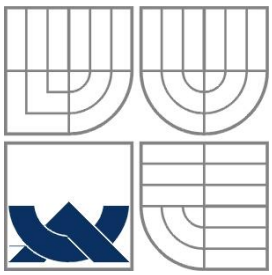


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH A REALIZACE DATABÁZE INFORMAČNÍHO SYSTÉMU PRO ZPRACOVÁNÍ RIZIK OBRÁBĚCÍCH STROJŮ

DESIGN AND IMPLEMENTATION OF DATABASE OF INFORMATION SYSTEM FOR PROCESSING OF MACHINE-TOOL RISKS

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **BC. STANISLAV ŠVOMA**
AUTHOR

VEDOUCÍ PRÁCE **ING. JIŘÍ KOVÁŘ**
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Stanislav Švoma

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Návrh a realizace databáze informačního systému pro zpracování rizik obráběcích strojů

v anglickém jazyce:

Design and implementation of database of information system for processing of machine-tool risks

Stručná charakteristika problematiky úkolu:

Informační systém pro zpracování rizik obráběcích strojů je založen na technologii firmy Microsoft. Datovou základnu tvoří databáze v SQL Serveru. Cílem diplomové práce je vytvořit
vhodnou strukturu databáze, procedury a C# adaptér.

Cíle diplomové práce:

1) Analyzujte požadavky na informační systém z hlediska databázové struktury
2) Navrhněte vhodnou strukturu databáze, procedur a C# adaptér
3) Ověřte funkčnost řešení na reprezentativním vzorku dat

Seznam odborné literatury:

[1] Kline, K., Gould, L., Zanevsky, A.; Transact-SQL Programming, O'Reilly, ISBN: 1-56592-401-0
[2] Brimhall, J., Dye, D., Gennick, J., Roberts, A., Sheffield, W.; SQL Server 2012 T-SQL Recipes, Apress,2012

Vedoucí diplomové práce: Ing. Jiří Kovář

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/2014.
V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D. doc.	Ing. Jaroslav Katolický, Ph.D.
Ředitel ústavu	Děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá analýzou, návrhem a realizací databázové struktury informačního systému, který je primárně určen pro zpracování rizik obráběcích strojů. Diplomová práce obsahuje rešeršní část, která obsahuje základní pojmy, přístupy a metody vývoje informačních databázových systémů. Prace dále obsahuje teoreticky návrh a praktickou realizaci zmíněného systému.

ABSTRACT

This thesis deals with analysis and design of a database structure for an information system, which is primary designed for processing of a machine-tool risk. The diploma thesis contains retrieval part, which describes concepts, approaches and methods of development of information database systems. The thesis contains theoretical concept and practical realization of database system, which is crucial purpose of this thesis.

KLÍČOVÁ SLOVA

Informační a databázový systém, UML, SQL, unifikovaný vývoj aplikaci

KEYWORDS

Information and database system, UML, SQL, Unified process

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že je tato diplomová práce mnou vypracovaná samostatně a všechny zdroje, prameny a literaturu, které jsem při vypracování použil nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

..... **.....**

V Brně dne podpis

BIBLIOGRAFICKÁ CITACE

ŠVOMA, S. *Návrh a realizace databáze informačního systému pro zpracování rizik obráběcích strojů.*
Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 54s. Vedoucí diplomové práce Ing. Jiří Kovář.

PODĚKOVÁNÍ

Tímto děkují svému vedoucímu práce Ing. Jiřímu Kovářovi za trpělivost, vstřícnost a cenné rady při tvorbě této diplomové práce.

Obsah:

1 ÚVOD

Tato diplomová práce se zabývá návrhem a realizací databáze informačního systému pro zpracování rizik obráběcích strojů. Hlavním účelem databázových systému je uchovávání velkého množství dat, které lze pak efektivně zpracovávat. Databáze mají využití například v knihovnických systémech, mezinárodních bankách a burzách, které zpracovávají statisíce transakcí denně.

Databáze se skládá z tabulek obsahující data. Tyto tabulky jsou mezi sebou propojeny vazbami, které společně s tabulkami tvoří strukturu databázového systému. Pro manipulaci s daty v rámci databáze se využívá specializovaných programovacích jazyku. V případě této diplomové práce se jedná o relační databázi, která pro manipulaci s daty využívá procedurální jazyk Transact-SQL.

Teoretickou náplní této diplomové práce je analýza a návrh databázové struktury. Analýza samotného vývoje je probrána v kapitole 3, která detailně rozebírá jednotlivé přístupy a metody pro tvorbu databází. Z popsaných metod pro řešení zmíněného informační systému byla vybrána iterační metoda UP (Unified Process). Tato metoda je rozdělena do pěti aktivit (požadavky, analýza, návrh, implementace a testování). Jednotlivé etapy vývoje jsou probrány v každé kapitole zvlášť. Na začátku jsou detailně rozpracovány požadavky na systém a jejich specifikace, tyhle požadavky představují co má systém dělat a nikoliv jak je má udělat. Tuto problematiku řeší kapitola 4. Další částí je analýza pro zpracování funkčních požadavků na informační systém. Dále byl navržen analytický diagram systému tzv. diagram tříd, který popisuje logickou strukturu informačního systému a zpracovává pohled ze strany samotných uživatelů. Touto problematika je popsána v kapitole 5 a použitý diagram tříd se vytváří pomocí modelovacího jazyka UML. V následující kapitole s názvem „Návrh informačního systému“ je analytický diagram tříd upřesněn do návrhového diagramu tříd. Diagram tříd není ve stádiu analýzy závislý na platformě, a proto je v rámci návrhu tento model přesně specifikován na základě použitých technologií. Jsou zde navrhnuta rozhraní zvyšující flexibilitu celého systému a všechny stávající asociace jsou upřesněny do rozšířených typů asociací tzv. agregací a kompozic tak, že je dosaženo databázové normalizace třetí formy a snížení nežádoucí redundance celého systému. Takto upravený model je pak přímo implementován.

Praktická část představuje pátou aktivitu metodiky UP a tou je samotná implementace návrhového modelu do relační databáze včetně vytvoření funkčních procedur za pomocí již zmíněného jazyka Transact-SQL a adaptéru pro platformu .Net v rámci jazyka C# pro nahrávání dat do databáze z .xlsx souboru. Na úplný závěr jsou zobrazeny i obrázky samotného webu pracujícího s vytvořenou databází.

2 MOTIVACE K PRÁCI

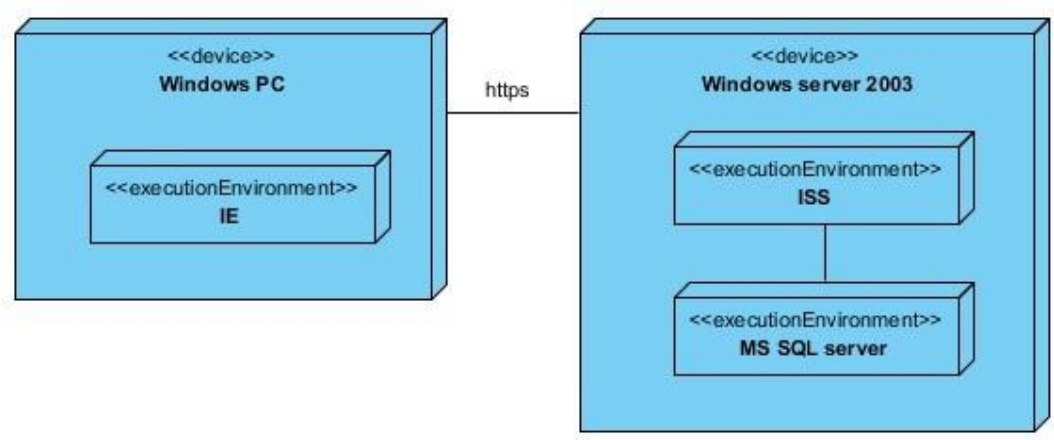
Náplní této práce je získat praktické zkušenosti s analýzou, tvorbou a návrhem informačního systému. K tomuto tématu lze přistupovat pomocí několika vývojových metod a jejich úspěšné zvládnutí dovoluje vyvinout fungující informační systém. Jednotlivé vývojové metody pracují s různými etapami zpracovávání systému, které v komerčním prostředí vykonává několik lidí současně.

Vyvíjena databáze musí uchovávat reálná data norem obráběcích strojů a nikoliv jen testovací data. Navržená databáze by měla být schopna data správně uchovávat a dokázat je specificky filtrovat a zobrazovat pro uživatele s různými oprávněními, proto je zapotřebí věnovat dostatek času jednotlivým částím vývoje informačního systému a všechny části správně probrat a definovat.

Reálné využití této práce je v přehledné správě dat, ke kterým bude možné přistupovat prakticky z jakékoliv části země.

3 ROZBOR INFORMAČNÍHO SYSTÉMU

Relační databáze je jádrem funkčního informačního systému. Databáze v této práci je vytvořena v prostředí Microsoft SQL Serveru, který pro komunikaci mezi tabulkami využívá jazyk Transact-SQL (T-SQL). Celá databáze je uložena na Windows Serveru, kde využívá Microsoft Internetovou Informační službu ve verzi 7. IIS7 dovoluje webovým návštěvníkům pomocí protokolu http přistupovat k datům informačního systému. Pro nahrávání dat lze použít adaptér, který je vytvořený v C#. Přehled nasazení je zobrazen na následujícím Obr. 1.



Obr. 1 Grafické znázornění samotného nasazení IS

- ***Relační databáze*** - první relační databáze byla definována již v roce 1970 matematikem Edgarem F. Coddem. Relační model, který vytvořil, vycházel z matematické relační algebry, kde jsou data ukládána do množin a jejich podmnožin. Koncem 70 let v laboratořích IBM vznikl nástroj pro komunikaci s relační databází, který se jmenoval SQL (StructuredQueryLanguage). Tento jazyk byl americkým institutem ANSI standardizován jako SQL – 86. Původní jazyk SQL byl definován pouze jako deklarativní, tedy založen na myšlence popsání úkolu (co se má udělat). Do dnešní doby byl SQL upravován a doplňován a od roku 1999 má již i procedurální rozšíření. Přes několikero standardizací se vyskytují i nestandardní procedurální řešení v závislosti na výrobci. V rámci této diplomové práce je využíváno procedurální řešení od Microsoftu, které se nazývá Transact-SQL [6] [1].
- ***T-SQL*** - doplňuje standard SQL o procedury, spouštěče, cykly, příkazy proměnné a další operace. [6]
- ***IIS7*** - (Internetová informační služba 7.0) je soubor programu na webovém serveru, pomocí kterých lze hostovat data na internet. Využívá Microsoft technologii xml, .net, activeX a podporuje ftp, http, smtp atd. [14].

3.1 Účel informačního systému

Hlavní účel vývoje informačního systému (dále jen IS) v této diplomové práci je usnadnění uchování konkrétních dat systému. IS umí uchovávat seznam strojů a seznam norem. Pojem zpracování rizik v této práci představuje normy zabývajícími se různými tolerancemi obráběcích strojů. Další rizika ovlivňující obráběcí stroje jsou obsaženy v zákonech,

směrnicích a nařízeních, které mají rozdílnou syntaxi než normy a v oblasti analýzy jsou v grafu zpracovány, přestože nakonec nebudou implementovány.

Jednotlivé normy, které IS obsahuje, jsou vydávány jednotlivými institucemi viz Tab. 1 a mohou být tříděny v systému různými způsoby např.: dle ISC kódu, kódu normy, označení atd. V případě této práce bude prováděno hlavní třízení do jednotlivých skupin pomocí předem daných kategorií.

Tab. 1 Přehled označení jednotlivých norem [16][19][20]

Skupina	**Význam skupiny**
ČSN	české technické normy původně představovalo československé normy.
ANSI	AmericanNationalStandardsOrganization) - Americká národní standardizační organizace
ISO	INTERNATIONL ORGANIZATION STANDARDIZATION (Mezinárodní organizace pro normalizaci)
EN	evropské normy
DIN	DeutscheInstitufurNormung
BS	British Standart
ETSI	EuropeanTelecommunicationsStandards Institute (Evropsky úřad pro telekomunikace)
IEC	InetrnationalElectrotechnicalCommission Mezinárodní elektrotechnická komise

IS bude mít rozdělený přístup na privátní a veřejné zobrazení. Každý přístup dovoluje zobrazovat různě obsáhlá data. Veřejná část bude umožňovat omezený přístup k datům bez přihlášení uživatele (pouze obecný seznam strojů a seznam norem). V privátní části bude mít uživatel po přihlášení přístup k detailnímu popisu norem a jejich tolerancím, které jsou rozděleny v závislosti na jejich kategoriích. V diplomové práci je definován požadavek na výslednou podobu norem. Základní rozdělení norem je podle kategorií (viz Tab. 2) a každá kategorie zobrazuje přehled obsažených norem, včetně jejich vlastností (viz *Tab. 3*). Přestože výsledná podoba bude ve výsledku stejná jako v tabulce, je zapotřebí analyzovat a navrhnout databázovou strukturu tak, aby byla odstraněna redundance a aby bylo možné jednotlivé normy efektivně spravovat. Dalším požadavkem na databázi je správa uživatelů.

Tab. 2 Kategorie a jejich rozdělení

Kategorie	**Popis**
Přesnost polohování v ose	*Přesnost a opakovatelnost nastavení…*
Přesnost kruhové interpolace	Zkouška určuje přesnost…
Zkoušky geometrické přesnost	Jedná se o normalizované soubory metod …
…	…

Tab. 3 přehled zdrojových dat pro databázi v kategorii: Zkoušky geometrické přesnosti

Norma	**Typ stroje**	**Omezení**	**Označení**	**Tolerance**	**Popis**	**Stav**
ISO 230-1:2012	Vše	Lineární osy	Straintess error	-	Chyba…	Platná
ISO 230-1:2012	Vše	Lineární osy	Angular error	-	Tři úhlové chyby…	Platná
ISO 230-1:2012	Vše	Rotační osy	Axial error	-	Axiální chyba…	Platná

3.2 Projektování informačního systému

Efektivní vývoj systému využívá vždy jednotlivé přístupy k projektování, jež jsou definovány v konkrétní metodě, která podobně jako metodiky stanovuje pravidla pro vývoj informačního systému. Nicméně na rozdíl od metodiky je metoda obecnější a nepřikazuje, pouze doporučuje směr vývoje.

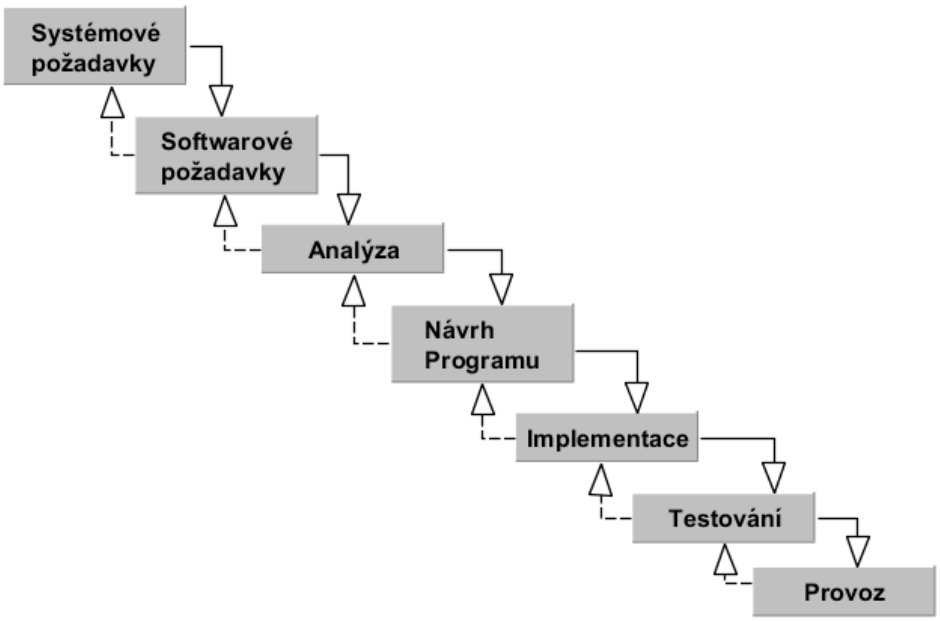
Pokud na začátku vývoje není dohodnut jednotný přístup k řešení problému, hrozí nebezpečí, že se začne vyvíjet systém za pomocí tzv. „anti"-metody TUNEL (občas definována jako průzkumná). Tato metoda ignoruje jakékoliv koncepce a doporučení. Zabraňuje znuvupoužitelnost stávajících projektu a řídí se pouze intuicí lidi pracujících na daném projektu. Metoda TUNEL je popisována, jako vstup projektů do černého tunelu. Na začátku projektu nejsou přesně definovány samotné cíle a celý vývoj představuje slepý průchod tunelem. Tento průchod si lze představit jako slepé narážení do stěn, neboli do slepých vývojových část, které se neměly vyvíjet. Tento postup se opakuje tak dlouho, dokud není nalezeno světlo na konci tunelu. Negativní důsledky zmíněné metody jsou především vysoká netransparentnost systému, složitost, redundance a časová náročnost. Tyto chyby zapříčiňují časté opravy samotného systému. Správným zvolením jediného modelu životního cyklu z následujícího seznamu se lze vyhnout zmíněné metodě TUNEL. V následujících podkapitolách bude popsáno několik modelů pro vývoj softwaru [1] [4] [9].

3.2.1 Vodopádový model

Vodopádový byl definován Winstonem W. Roycem v roce 1970 představuje nejzákladnější typ softwarového vývoje a samotný vodopádový model je rozdělen do sedmi základních etap.

- První etapa: systémové požadavky
- Druhá etapa: softwarové požadavky
- Třetí etapa: analýza
- Čtvrtá etapa: návrh programu
- Pátá etapa: implementace
- Šestá etapa: testování
- Sedmá etapa: provoz

Celý princip spočívá z přechodu od jedné etapy ke druhé při splnění jediné podmínky, že předchozí etapa je úplná a kompletně zdokumentovaná. Podobně jako vodopád, který může téci pouze jedním směrem, tak i zde se nepočítá s vrácením k předchozí etapě, nedochází tedy k cyklení. Nicméně tento základní princip není produktivní a slouží jako základní kámen pro ostatní modely, které jsou specificky rozšířeny např.: *V-sharped model,* který je doplněn různými testy a verifikaci během vývoje nebo inkrementální (multi-vodopádový) kde samotné etapy vodopádového modelu jsou rozděleny do dalších menších vodopádů. Grafické znázornění vodopádové metody je zobrazeno na (Obr. *2*) [1][13].



Obr. 2 *Winstonův vodopádový model [13]*

3.2.2 Prototypový model

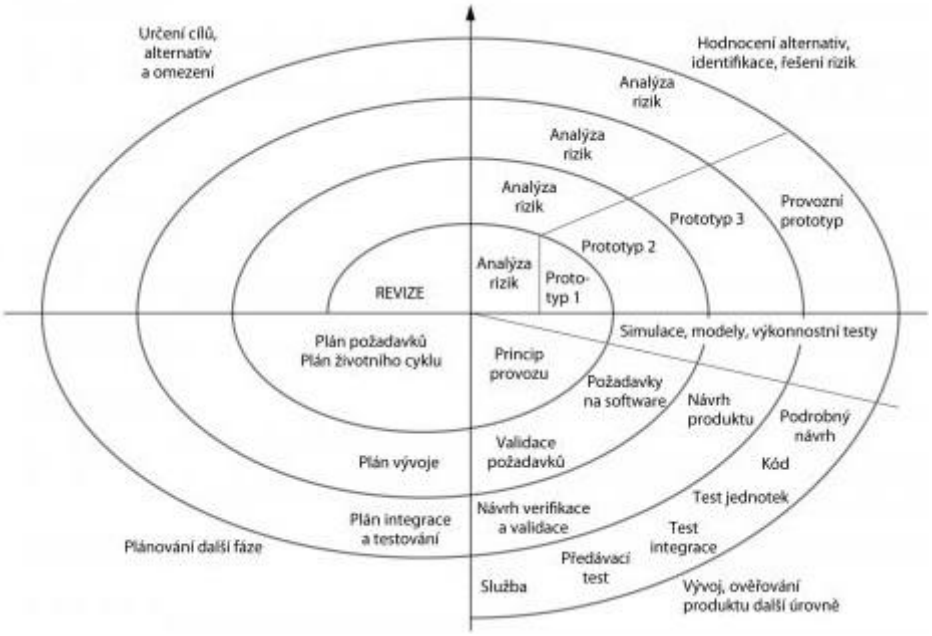
Prototypový model představuje inkrementální proces, v rámci této metody je vytvořen „prototyp“ který je předán zadavateli. Zadavatel tento prototyp testuje a následně zasílá vývojovému týmu zpětnou vazbu. Na rozdíl od metody „TUNEL“ jsou všechny připomínky důkladně analyzovány. Po důkladné analýze se přechází do etapy návrhu systému. Tento proces se postupně opakuje, dokud není vytvořen výsledný model informačního systému [1].

3.2.3 Spirálový model

Spirálový model byl definován Barrym Boehmem v roce 1988 a kombinuje prvky předchozích modelů. Celý proces vývoje je zobrazen pomocí spirály a nikoliv pomocí jednosměrného přechodu. Samotná spirála je iterativní a rozdělena do čtyř kvadrantů s různými vývojovými fázemi, ty se postupně opakují a tím postupně dochází k podrobnější analýze, návrhu i implementaci.

- *První kvadrant* definuje cíle celého projektu, vytváří se detailní plán a definují se alternativní omezení, které mohou vyplynout z analýzy rizik.
- *Druhý kvadrant* provádí hodnocení alternativ, identifikuje a analyzuje rizika spojená s tvorbou systému.
- *Třetí kvadrant* představuje samotný vývoj systému, jeho testování a kontrolu.
- *Čtvrtý kvadrant* stanovuje nový plán pro další iteraci

Opakování vývojových fází počítá se změnami v systému, které jsou způsobeny rizikem vývoje. Boehmův model tedy zavádí aktivity pro řízení rizik kvůli jejich omezení. Boehmova spirála je zobrazena na (Obr. 3) [1][4].



Obr. 3 Boehmuv spirálový model [4]

3.3 Přístupy pro vývoj informačního systému

Základní přístupy k vývoji a analýze IS jsou dva. První přístup, který se zabývá strukturovanou analýzou a návrhem, vznikl koncem 70 let 20. století. Druhý mladší přístup se začal rozvíjet v polovině 80 let 20. století a zabývá se objektově orientovanou analýzou a návrhem. Oba dva přístupy se liší v metodách, které používají a lze je navzájem kombinovat. Takovému kombinovanému přístupu se obecné říká hybridní přístup.

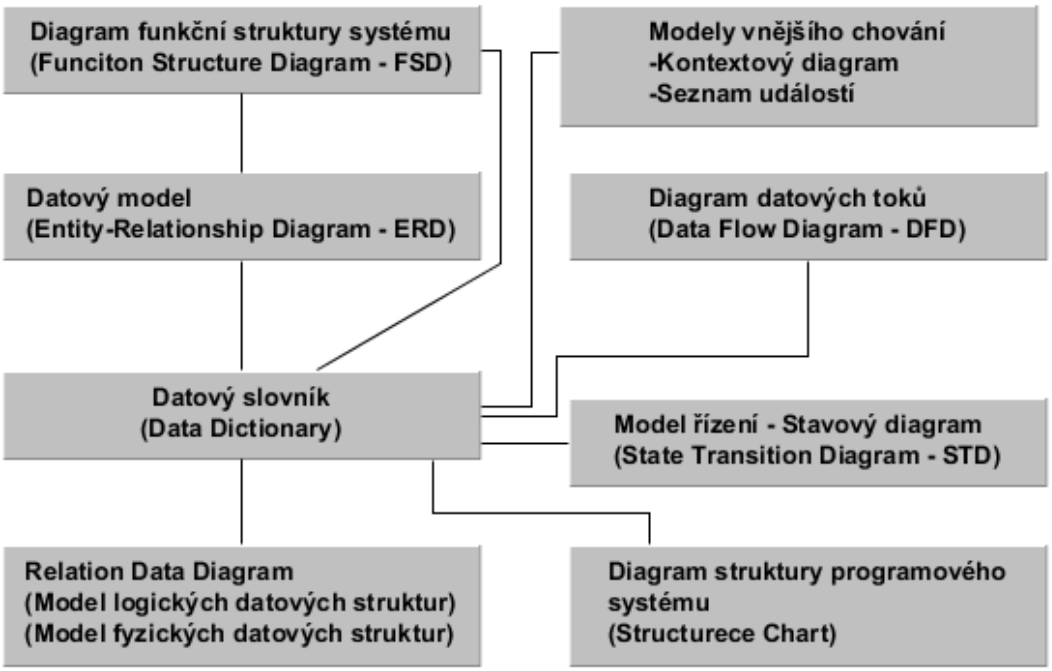
V každém přístupu se následně modelují diagramy od nejvyšší úrovně abstrakce až po nejnižší, z toho důvodu, aby pomohly překlenout sémantickou mezeru. Díky tomu je možné nalezení shody či neshody dřív, než je vytvořen kompletní informační systém.

Modely nejvyššího stupně abstrakce se nazývají analytické modely a představují vysoce abstraktní požadavky na vyvíjený informační systém, např.: systém zpracovává normy, uživatel se přihlašuje do systému, administrátor aktivuje účty atd. Tyto modely jsou tedy nezávislé na použitých technologických nástrojích a neobsahují implementační podrobnosti. V praxi se o nejvyšší úrovně abstrakce starají analytici a konzultanti.

Druhá abstraktní úroveň představuje modelování návrhu. Návrh přebírá analytický model a ten je upřesňován pro použití na konkrétní platformě. Výsledek druhé abstraktní úrovně je pak předán samotným programátorům, kteří provedou praktikou realizaci návrhu. Programování se tedy považuje za poslední nejnižší úroveň abstrakce [1].

3.3.1 Strukturální přístup

Strukturální přístup je definován tím, že vytváří samostatnou strukturu pro uložená data a samostatnou strukturu pro procesy pracující s daty. Strukturální přístup využívá modelovací nástroje, které jsou rozděleny na statické modely a modely dynamiky systému. Statické modely jsou na následujícím Obr. *4* zobrazeny vlevo a modely dynamiky vpravo.



Obr. 4 *Schéma modelu strukturovaného přístupu [1]*

Znázorněné diagramy je možné vytvářet v různých úrovních abstrakce. V rámci strukturálního přístupu se vyvíjí jednotlivé diagramy zvlášť, ale zároveň musí být všechny diagramy konzistentní, aby vytvářely jeden funkční celek informačního systému. Proto se musí takové diagramy ERD a DFD vyvíjet paralelně. Popis všech prvků je pak slovně definován v datovém slovníku, včetně jejich spojení.

- Modely vnějšího chování -Vývoj IS zpravidla začíná návrhy modely vnějšího chování, konkrétně kontextovým diagramem, který zobrazuje hranice systému a popisuje jednotlivé interakce s okolím pomocí procesů na nejvyšší úrovni abstrakce. Druhý model vnějšího chování je seznam událostí a jak samotný název napovídá, tak v rámci tohoto modelu jsou definovány jednotlivé události, které mohou vzniknout při používání systému. Události se v modelu chování definují jako T, F, C-události (Flow, Temporal, Control).
- FSD - (Diagram funkční struktury) je statický model systému, který vyjadřuje organizační strukturu a je zobrazován pomocí stromové struktury shora dolů, kde kořen stromu představuje systém, uzly zobrazují jednotlivé podsystémy a listy jejich funkce. Jednotlivé funkce pak mohou být procesní, dialogové nebo řídící.
- DFD – (Diagram datových toků) je dynamicky model a zobrazuje vstupy a transformované výstupy v samotném systému v nižší abstrakční úrovni než modely vnějšího chování. Všechny vstupy lez pak dekomponovat až na nejzákladnější procesy systému. DFD model se vytváří za pomoci grafického návrhu, kde každý prvek musí mít vysvětlující popis. Přestože DFD znázorňuje samotné procesy, tak zde není zobrazováno jejich časové uspořádání a ani pořadí jednotlivých datových toků.
- ERD – (Entity-relationship diagram) se obecně využívá pro logický návrh a reprezentaci dat v databázi na vyšší úrovni abstrakce. ERD je složen z entit (objektu reálného světa) a atributů (vlastnosti a charakteristiky každé entity) a jednotlivé entity jsou mezi sebou navzájem propojeny tzv. vztahy. U jednotlivých

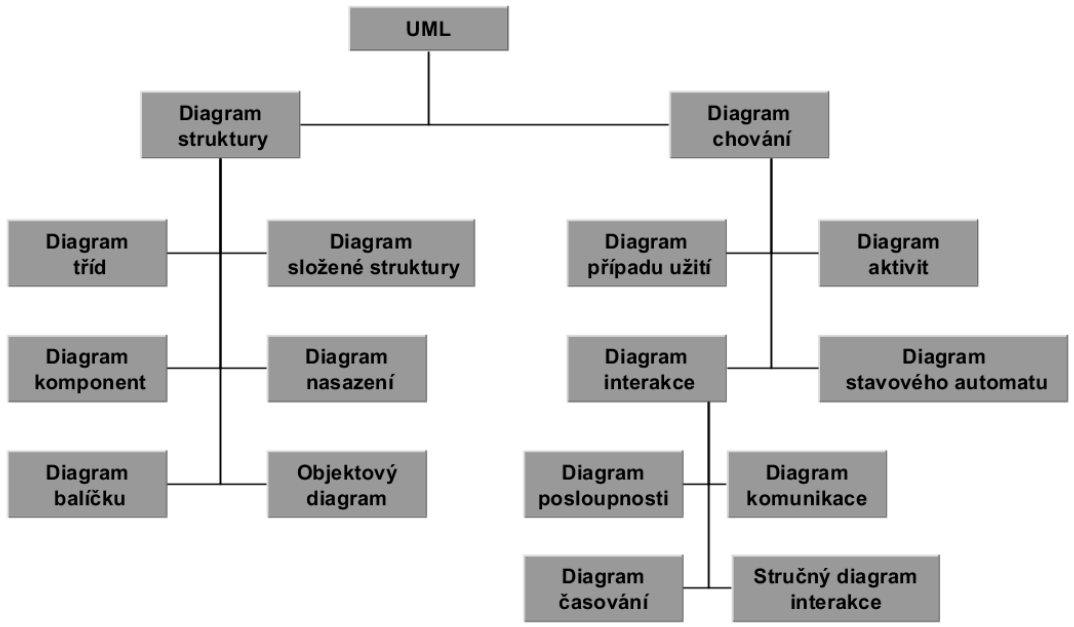
vztahů je definována kardinalita vztahu (1:1, 1:N, N:1, N:M), která zobrazuje kolik entit má daný vztah. Při snižování abstrakce u ERD se přesouváme do modelu RDD.

- RDD – (Relation Data Diagram) Tyto modely logických a datových struktur stanovují pomocí normalizace databázové normální formy. A zajišťují integritní omezení za pomocí primárních klíčů (integrita entity), cizích klíčů (integrita referenční) a doménová integrita. Při přechodu z modelu logických datových struktur do modelu fyzických datových struktur se zároveň mění i používaná terminologie názvu u jednotlivých modelů.
- STD - (State Transition Diagram) Ve stavovém diagramu se modeluje časová závislost a jejich posloupnost. Stavový diagram je definován pomocí uzlu a hran, kde uzly definují stavy systému a hrany jsou jejich přechody. Zároveň obsahují i skladiště události, které mění stav systému až po splnění několika podmínek.
- SCH – (Structure Chart) Diagram struktury se zobrazuje stromovou strukturou podobně jak FSD. Kořen SCH představuje hlavní programový modul a uzly jsou jednotlivé sub moduly. Jednotlivé návrhy lze vytvářet za pomocí tří technik
  a) funkční dekompozicí
  b) analýzou datových toků
  c) analýzou datových struktur

3.3.2 Objektově orientovaný přístup

Tento přístup vytváří struktury pomocí objektů, které mají zároveň atributy a operace definované přímo v objektu. Jednotlivé objekty pak mezi sebou komunikují na základě impulsu. Standardizovanou metodou pro tvorbu objektově orientovaného přístupu se stal jazyk UML. Samotný jazyk představuje pouze vizuální syntaxi pro modelování různých diagramu a není vázán na výše zmíněné životní cykly a ani na metody zmíněné v další kapitole.

První verze jazyka UML byla standardizována v roce 1997 společnosti OMG (Object Management Group) a od té doby je UML neustále vyvíjeno a rozšiřováno. V roce 2005, byla vypuštěna verze 2.0 a v dnešní době je už verze 2.5 beta. Jazyk UML, který je tedy vyvíjen již 17 let představuje velice vyspělý modelovací jazyk. UML rozděluje své diagramy na diagramy struktur, popisující všechny důležité objekty a jejích propojení mezi sebou a diagramy chování, zobrazující interakce v systému, které jednotlivé objekty vykonávají mezi sebou. Jednotlivé diagramy jsou mezi sebou podobně závislé, jako ve strukturálním přístupu. Detailní rozdělení všech diagramu jazyka UML je znázorněno na (Obr. 5) [1][3].



Obr. 5 Zobrazení všech diagramu v UML 2.4 [3]

Nejčastěji používané diagramy jazyka UML jsou následující:

- *Diagram případu užití* – Hlavním smyslem diagramu případu užití je nalezení funkčních požadavku. Představují vysokou úroveň abstrakce a dávají odpověď na otázku „CO“ systém dělá, nikoliv na to „JAK“ to systém dělá. Modely případu užití představují interakce programu s tzv. aktérem tj. uživatelem nebo jinou částí programu, která něco vykonává. Každá část se navrhuje separátně a představuje konkrétní funkčnost.
- *Diagram tříd* – představuje nejkomplexnější a nejvyužívanější diagram v celém jazyku UML. Diagram tříd je modelován pomocí tříd a relacemi mezi nimi. Samotné třídy uchovávající data množiny objektů se společnými atributy a operacemi. Relace pak zobrazují, jak spolu jednotlivé třídy komunikují. Diagram tříd se využívá v etapě analýzy i etapě vývoje informačních systémů.
- *Diagram balíčků* – balíčky se používají pro přehlednější uspořádání větších systému a představuje složku, která uchovává společné prvky jednotlivých modelů.
- *Diagram nasazení* – zobrazuje skutečné nasazení informačního systému, včetně softwarových komponent, které jsou zapotřebí k jeho spuštění. Diagram nasazení je zobrazen hned na začátku této kapitoly viz Obr. 1.

3.4 Metody pro vývoj informačních systémů

K jednotlivým přístupům pak můžeme v rámci softwarového inženýrství přistupovat pomocí tradičních metodik nazývané též rigorózní metodiky, které představují historicky starší metodiky vývoje a dbají především na podrobné rozpracování analýzy a vývoje. Na začátku vývoje je daný jasný cíl a všechny jednotlivé části jsou podrobně zdokumentovány, přičemž se zde nepočítá s častou změnou systému. Vytváří se obsáhlá dokumentace a popis jednotlivých částí. Pro tradiční metodiky se využívají především modely vodopádové a

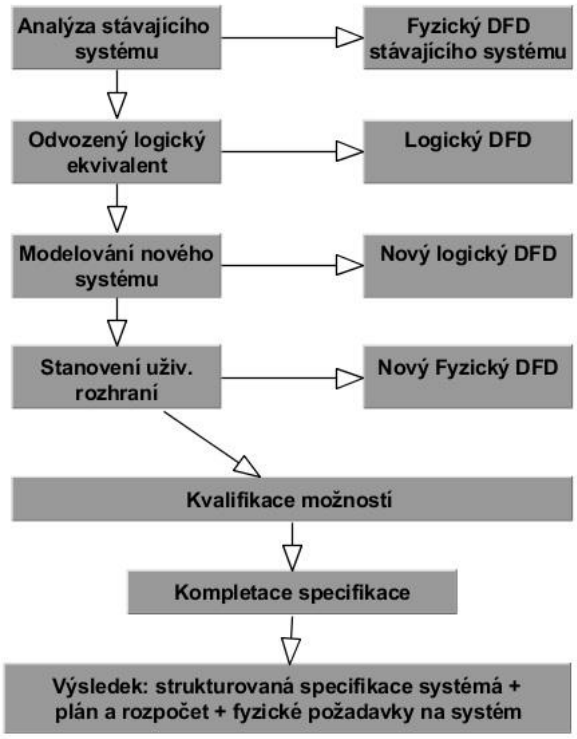
inkrementální. Anebo lze použít agilní metodiky, u kterých se počítá s častou změnou. Nedělá se zde tak podrobný rozbor jako u tradičních metodik. Celý princip je založen na vysoké komunikaci se zadavatelem informačního systému, kterému jsou v co nejkratších časech dodávaný nové prvky systému. Systém je tedy rozdělen do malých vývojově nezávislých iterací, které jsou co nejrychleji vyvíjeny. Podle nového požadavku je pak vytvořená iterace znovu analyzována a případně znovu vyvíjena. Tento proces trvá tak dlouho, dokud není zadavatel spokojený. Agilní metodiky se tedy nesoustředí na obsáhle vývojové diagramy, ale vyvíjejí se zde momentálně potřebné části. V agilních metodikách se především setkáme se spirálovými a prototypovými typy vývoje. Nejznámější a nejpoužívanější metody jsou zobrazeny v Tab. 4.

Tab. 4 Přehled metod dle přístupu

Strukturované	Objektové/Tradiční	Objektové/Agilní
DeMarcova metoda	Boochova metoda	SCRUM
Gane/Sarsonova metoda	Metoda OMT	Extrémní programování
Yourdonova analýza	Metoda UP	

3.4.1 DeMarcova metoda

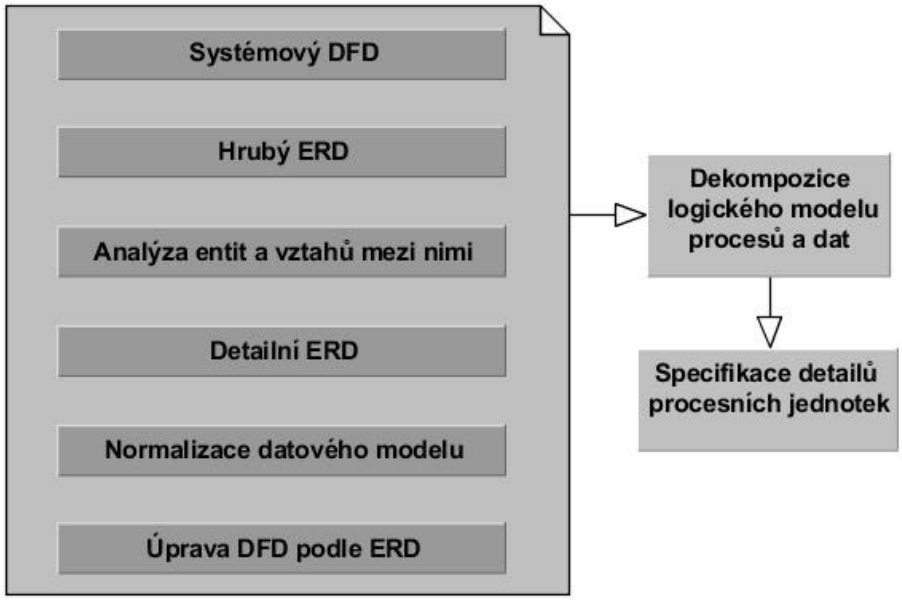
Tato metoda byla definována Tomem DeMarcem v roce 1979. Základem této metody je rozdělení systému na jednotlivé části a grafické vytvoření DFD od nejvyšších úrovní abstrakce až po nejnižší úrovně, kde jednotlivé prvky DFD jsou detailně popisovány v datovém slovníku. Tato metoda je funkčně zaměřená a vůbec nepoužívá ERD. Grafické schéma postupu je zobrazeno na (Obr. 6). [1]



Obr. 6 Metoda DeMarco[1]

3.4.2 Gane/Sarsonova metoda

Byla vyvinuta Chris Ganem a Trish Sarsonem chvíli po DeMarcovi. Hlavním rozdílem mezi DeMarcovou a Gane/Sarsonovou metodou je, že využívá datově orientovaný přístup a přidává ERD. Začátek vývoje opět začíná vytvořením DFD a následně je vytvořen ERD, který je důkladně analyzován, upřesněn a normalizován. Podle výsledného ERD je pozměněn původní DFD. Schéma je zobrazeno na Obr. 7 [1]



Obr. 7 Metoda Gane/Sarson [1]

3.4.3 Yourdonova strukturovaná analýza

Yourdonova analýza pochází z roku 1989 a představuje sjednocení předchozí metod v jeden celek. Vývoj pomocí této metody je rozděleno do několika etap, kde výsledkem celé analýzy je vytvoření esenciálního modelu (esence = podstata).

Samotný esenciální model je zmapován pomocí několika modelů. První model prostředí, popisuje jednotlivé situace, které mohou nastat a jak na ně systém bude odpovídat. Pro model prostředí se používá seznam událostí a kontextový diagram. Druhý model chování systému zobrazuje samotná data a procesy uvnitř systému. Celkový model je pak definován různými diagramy DFD (FSD), ERD, datovým slovníkem a STD, které musí být navzájem konzistentní.

Druhou etapou je vytvoření uživatelského implementačního modelu. V této etapě je esenciální model upravován až na implementační úroveň a je zároveň navrhnuto uživatelské rozhraní.

Třetí etapa využívá uživatelsky implementační model pro vytvoření návrhu struktury programového systému [1][4].

3.4.4 Boochova metoda (OOD)

V roce 1991 byla publikována metoda OOD (Object Oriented Design) vyvinuta Grady Boochem [1]. Metoda pracuje iterativním a inkrementačním přístupem a tato metoda začíná pracovat i s diagramem tříd, který je později definován v upravené formě jazyka UML. Jednotlivé etapy jsou rozděleny do tří částí:

- *Analýzy* – v rámci analýzy jsou definován třídy a objekty.
- *Návrhu* – samotný návrh se zabývá jednotlivými vztahy a relacemi. Pro diagramy struktury používá diagram tříd a u diagramů chování pracuje s diagramy interakce a stavovými diagramy.
- *Implementace* – představuje samotné nasazení systému.

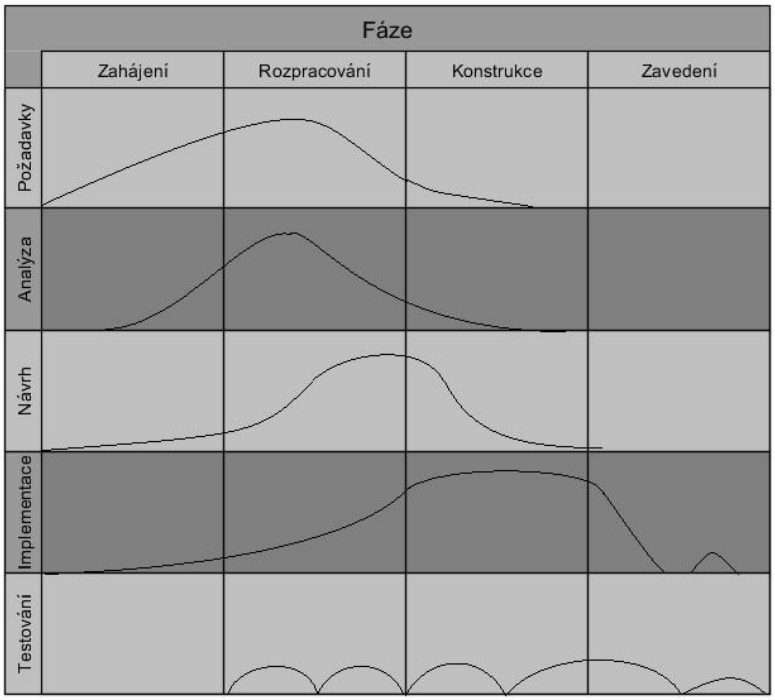
3.4.5 OMT metoda

Vznikla v roce 1991 a jejím autorem je James Rumbaugh [1]. Základ metody vychází z ERD tedy ze strukturovaného přístupu a přidává objektové prvky. Metoda pracuje s životním cyklem vodopád a je rozfázovaná do následujících čtyř fází

- *Analýza* – během analýzy jsou definované uživatelské požadavky, objektový model, DFD a v rámci diagramu chování seznam událostí a stavové diagram.
- *Systémový návrh* – představuje rozbor systému na jednotlivé subsystémy a jejich detailní popis a správu celého systému.
- *Objektově orientovaný návrh* – optimalizuje a upřesňuje jednotlivé modely z etapy analýzy na základě výsledků systémového návrhu.
- *Implementace* – systém se programově nasazuje

3.4.6 Metoda UP (Unified Process)

První obrysy vznikly již v roce 1967 v Ericssonově modelu. Ericssonův model pracuje s myšlenkou rozložení jednoho velkého systému, na malé dílčí součásti, které jsou navzájem propojené, stalo se tak z jednoduchého důvodu, kdy menší část systému se definují lépe než celek. Tato metodika vznikla v roce 1999 a byla vytvořena autory jazyka UML. Metodika UP je tedy založena na iterativním a přírůstkovém procesu vývoje aplikací, kde každá iterace představuje určitou část finálního systému. Skládá se z pěti pracovních postupů (požadavky, analýza, návrh, implementace a testování), jež popisují jak dospět konečného výsledku a čtyř fází (zahájení, rozpracování, konstrukce a zavedení), ve kterých je zobrazená náplň každého pracovního postupu. Schéma fází a postupu je zobrazeno na (Obr. 8) [3].



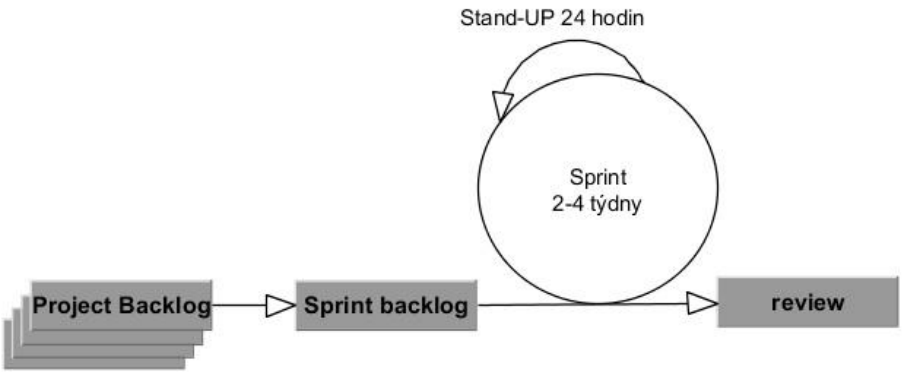
Obr. 8 Schéma fází metodiky UP [3]

3.4.7 Metoda SCRUM

Spadá do agilní metodiky a představuje metodu, která pracuje s menším týmem vývojářů. Na začátku celého procesu jsou definovány nestrukturované požadavky na systém, které se zaznamenávají v tzv. backlogu. Jednotlivé požadavky jsou následně rozděleny do menších backlogu a přiděleny jednotlivým iteracím neboli sprintům.

Sprint začíná fází planning, kde jsou z backlogu vybrány požadavky podle jejich priority a rozpracovány v rámci jednoho sprintu. Na základě požadavku je definované časové rozmezí (několik týdnů), které je opakovaně kontrolováno stand-upy (porady, ve kterých je shrnutí postupu od poslední porady). Celý sprint je zakončen fází review, kde je vytvořený funkční požadavek předveden stakeholderum. Po ukončení sprintu se provádí retrospektiva, za účelem vylepšení stávajícího požadavku a zhodnocení průběhu, zdali vše bylo splněno v řádném termínu

Tento postup se opakuje, dokud nejsou splněny všechny požadavky. Jednoduchý přehled je zobrazen na (Obr. 9) [12].



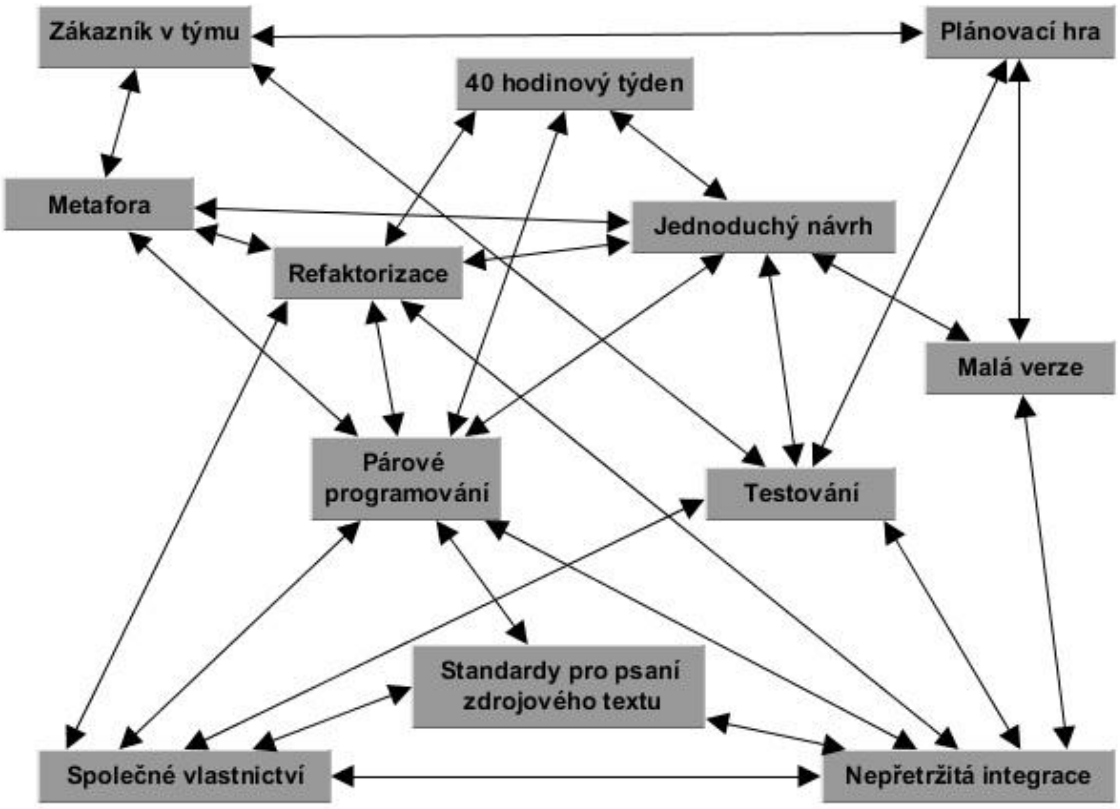
Obr. 9 Grafické znázornění metody SCRUM [12]

U metody SCRUM jsou přiděleny osobám následující role.:

- *Stakeholders* – koncoví zákazníci, kteří budou systém používat
- *Product owner* – přiřazuje jednotlivým požadavkům priority a je zodpovědný za výsledek celého projektu.
- *Scrum Master* – projekt manažer, který je zodpovědný za jednotlivé sprinty.
- *Team* – představuje všechny ostatní, kteří se podílí na vyvíjení systému.

3.4.8 Metoda extrémního programování (XP)

Bylo definováno Kentem Beckem a pracuje úplně s jinými přístupy než všechny ostatní metody a je extrémně zaměřeno na samotné programování. Všechny postupy jak návrhy a programování se píše ve zdrojovém textu. Extrémní programování (dále jen XP) předpokládá časté změny v systému, a proto pracuje s velice krátkými iteracemi. Programátoři v rámci XP pracují v malém týmu minimálně dvou osob. Tyto dvě osoby pracuji vždy společně z důvodu okamžité revize druhou osobou. XP provádí analýzu požadavku zákazníka, přes tzv. testy. Samotného zákazníka vyžaduje XP vždy v týmu, aby byl systém rychle vyvíjen a zabránilo se případné reklamaci. Vývoj pomocí XP předpokládá paralelním dodržování postupů, které jsou zobrazeny na (Obr. 10) [8].



Obr. 10 Prolínání postupu v metodě XP [8]

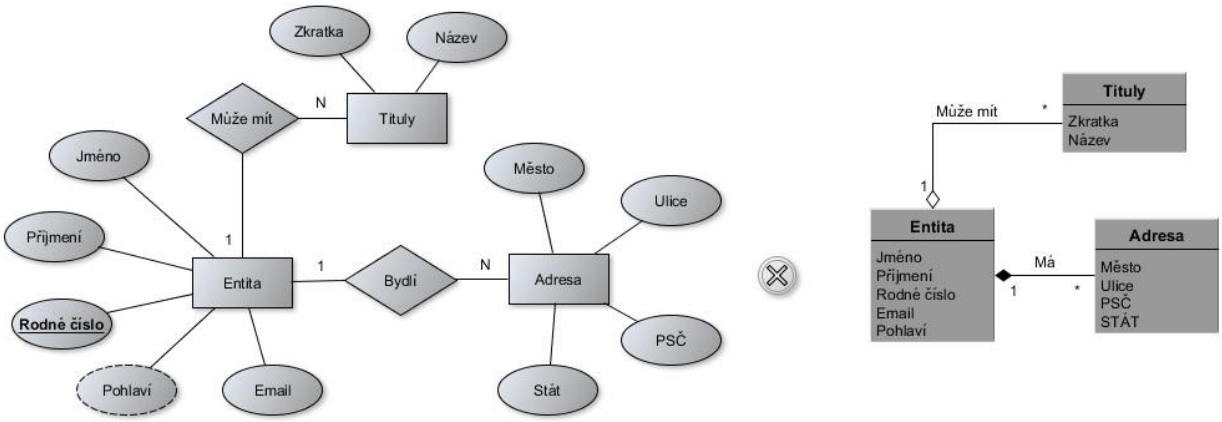
3.5 Výběr metody pro vývoj informačního systému

Výběr správné metody se odvíjí od požadavku na systém, přístupu, velikostí týmu a znalostí. Přestože byly v této práci zmíněny i populární agilní metody, pro vývoj databáze obsažené v této prací jsou nevhodné. Převážně z důvodu velké součinné orientace na vícečlenný tým.

Z hlediska přístupu se dají pro relační databázi využít jak metody strukturované, tak metody objektově orientované. Ze strukturovaných metod je nejvýhodnější použití Yourdonovy metody. Jedná se o nejmladší strukturovanou metodu a dovoluje využít nejvíce nástrojů pro vývoj IS. Za objektově orientovanou metodu je nejvýhodnější použít metodu UP. Tato metoda sjednocuje nejpoužívanější fáze vývoje a nabízí využití moderního jazyka UML.

Pro vývoj informačního modelu byla zvolena metoda UP, která nám dovoluje bezplatně využít široké možnosti jazyka UML pro hledání požadavku, analýzy a návrhu. Hlavní důvod je možnost použít objektový diagram tříd, který poskytuje přehlednější syntaxi než ERD a navíc jednotlivé třídy obsahují i seznam operací. Díky tomu nám jeden diagram UML dovoluje sloučit dva strukturální diagramy do sebe konkrétně ERD a DFD. Navíc jednoduše zobrazuje i jednotlivé požadavky na systém pomocí USE CASE modelu. Graf UML a ERD v základní syntaxi je zobrazen na Obr. 11.

Implementace databázové struktury bude dle zadání využívat MS-SQL s procedurálním rozšířením T-SQL a samotná komunikace jednotlivých částí databáze a webu, bude probíhat pomocí ASP.NET, který dovoluje vyvíjet a spravovat dynamický systém.



Obr. 11 Základní syntaxe ERD (vlevo)[10] vs. UML diagramu tříd (vpravo)[3] [5]

4 POŽADAVKY NA INFORMAČNÍ SYSTÉM

Dle metody UP i v ostatních metodách se v první etapě vývoje IS postupně zjišťují samotné požadavky na IS. Ne všechny požadavky na systém lze zobrazit či jednoznačně určit v případech užití nebo UML diagramu, ale počítá se s nimi již při návrhu. Typickým případem jsou tzv. nefunkční požadavky (převážné omezení systému odezvy, stabilita atd.), kdežto případy užití zobrazují pouze funkční požadavky (co systém děla) a které byly postupně zaznamenány. Získávání jednotlivých požadavku se provádí pomocí tří postupů

1. Konzultace
2. Dotazníky
3. Dílna požadavků

V rámci této práce probíhali jednotlivé konzultace, ve kterých byly nadefinovány obecné požadavky na systém. Po skončení konzultace byly jednotlivé případy analyzovány a navrhnuty první případy užití. Jednotlivé případy se navrhují tak aby dali odpověď na otázku „CO“ systém dělá, nikoliv na to „JAK“ to systém dělá. Modely případu užití představují interakce programu s tzv. aktérem tj. uživatel nebo jiná část programu. Každá část se navrhuje separátně a představuje konkrétní funkčnost. Celkový seznam všech vytvořených funkčních případů užití je zobrazen v Tab. 5 a název případu užití je psán velbloudí notací.

Tab. 5 Přehled případu užití

ID	Název	ID	Název
UC01	Přihlášení	**UC16**	VyhledatUživatele
UC02	UzamknutíÚčtu	**UC17**	ZměnitStavÚčtu
UC03	Registrace	**UC18**	ZobrazitDetailÚčtu
UC03.1	NeplatnéÚdaje	**UC19**	ZobrazitDetailUživatele
UC03.2	Storno	**UC20**	ZměnitÚdajeUživatele
UC04	ZobrazitProfil	**UC20.1**	NeplatnéÚdaje
UC05	EditovatOsobníÚdaje	**UC21**	SmazatUživatele
UC06	ZměnitKontakt	**UC22**	ZobrazitVšechnyUživatele
UC07	ZměnitHeslo	**UC23**	FiltrovatSeznamUživatelů
UC08	ProhlížetStroje	**UC24**	EditovatNormu
UC09	TříditPodleKatgorií	**UC25**	PřidatNormu
UC10	Tisknout	**UC25.1**	PřidatNormu:Adaptér
UC11	OvěřitUživatele	**UC25.2**	PřidatNormu:KontrolaFormátu
UC12	VyhledáváníNorem	**UC26**	OdebratNormu
UC13	ProhlížetNormy	**UC27**	PřidatStroj
UC14	VyhledáváníStrojů	**UC28**	EditovatStroj
UC15	AktivovatÚčet	**UC29**	OdebratStroj

Nefunkční požadavky představují omezující podmínky, vycházející z použitých nástrojů, jako je tabulkový procesor (MS-SQL), použití jazyku T-SQL a odstranění redundance v případě norem.

4.1 Modelování případu užití

Případy užití se modelují do grafu, který přehledně zobrazuje propojenost jednotlivých aktérů s konkrétním případem užití a zároveň lze použít i textový rozbor pro každý případ užití, kde je zobrazen scénář představující posloupnost jednotlivých kroků samotného případu užití. Příklad textového rozboru je znázorněn v Tab. 6 a Tab. 7. Jednotlivě zpracované případy užití lze nalézt v příloze diplomové práce.

Tab. 6 Textový rozbor případu užití: Přihlášení

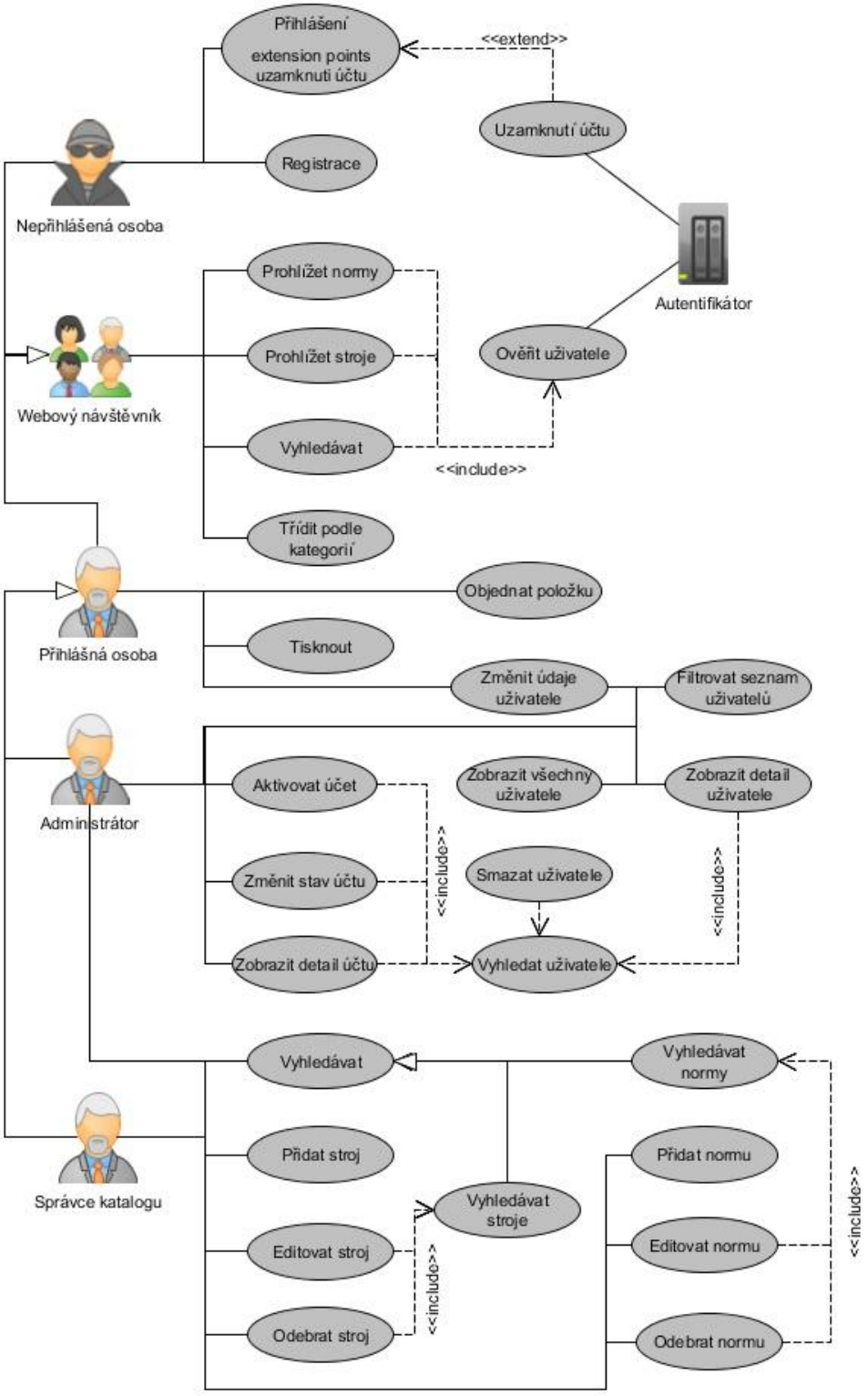
Případ užití: Přihlášení	
ID:	UC01
Stručný popis:	Systém přihlásí uživatele do systému
Primární aktéři:	Nepřihlášený uživatel
Vedlejší aktéři:	Žádný
Vstupní podmínky:	Žádné
Hlavní Scénář:	1. Webový uživatel zadá příkaz „Přihlásit“ 2. Systém požádá o login a heslo 3. **if** login nebo heslo jsou neplatné **místo rozšíření**: uzamknutí účtu 3.1. Systém požádá o zopakování loginu a hesla 3.2 ověření údajů **end if** 4. systém přihlásí uživatele
Výstupní scénáře:	Uživatel je přihlášen k systému
Alternativní scénáře:	1. Alternativní scénář začíná 3. krokem hlavního scénáře 2. Systém informuje uživatele, že zadal neplatné jméno nebo heslo

Tab. 7 Textový rozbor případu užití: Uzamknutí účtu

Případ užití: Uzamknutí účtu	
ID:	UC02
Stručný popis:	Účet se uzamkne, jakmile se zadá 5x špatně pin
Primární aktéři:	Autentifikátor
Vedlejší aktéři:	Žádný
Vstupní podmínky:	Uživatel zadává špatné heslo
Hlavní Scénář:	1. systém kontroluje pokusy neplatných přihlášení 2. u 5. pokusu zamkne účet
Výstupní scénáře:	Zamknutí účtu

Grafický model je celý zobrazen na Obr. 12. Při modelování případů užití je

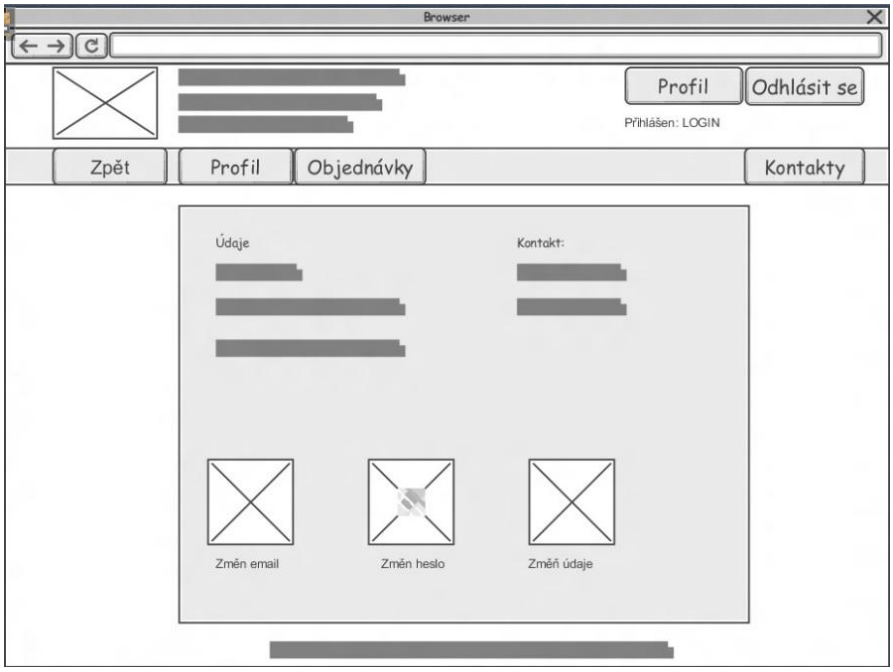
nežádoucí modelovat za pomocí funkční dekompozice. Smyslem je zachytit požadavky systému a proto by obecné třídy vytvořené pomocí funkční dekompozice nepřinesly nové poznatky do navrhovaného systému. Nicméně i přesto zde není hierarchie úplně zakázána, ale její hloubka by neměla přesahovat druhou úroveň.



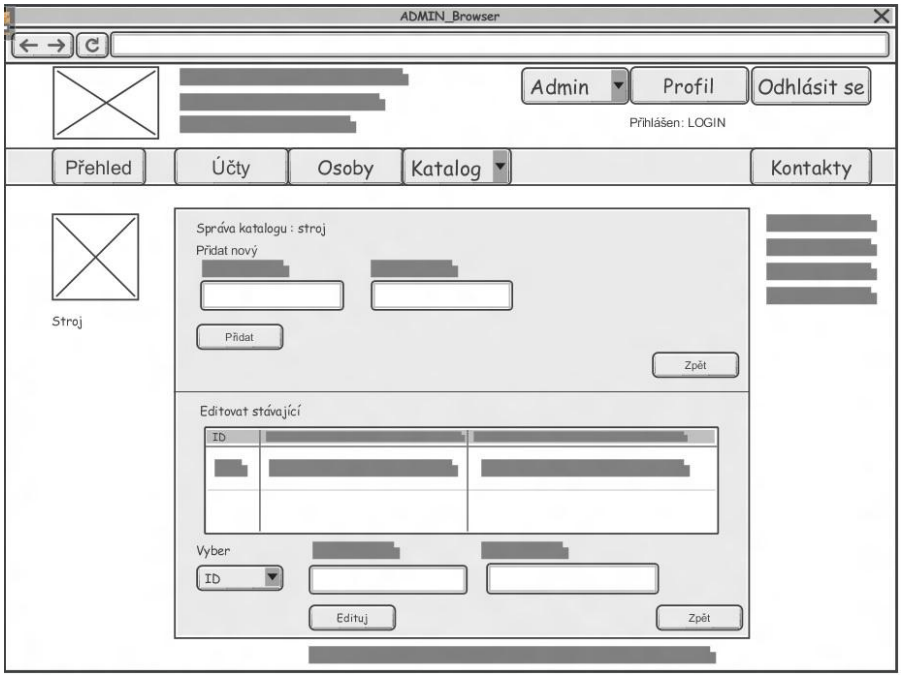
Obr. 12 Model případu užití

4.2 Wireframe

Představuje první náhled výsledného produktu a vytváří se pro lepší pochopení obsahu budoucích webových stránek. V důsledku takto vytvořenému návrhu lze upřesnit nebo pozměnit jednotlivé případy užití. Jednotlivé wireframy jsou opět uložené v příloze DP a pro ilustraci jsou zde uvedené pouze dva (viz Obr. 13 a Obr. 14).



Obr. 13 Wireframe zobrazující správu přihlášeného uživatele



Obr. 14Wireframe zobrazující editaci a přidání stroje přes webové rozhraní

5 ANALÝZA INFORMAČNÍHO SYSTÉMU

Analýza IS pomocí metody UP představuje základní tvorbu analytické modelu z pohledu jeho chování. U samotné etapy analýzy dochází k prolínání s etapou požadavku, proto mohou být i v rámci vytváření analytického modelu zachyceny nové požadavky nebo upraveny stávající. Výsledkem analýzy je model, který stále nedefinuje jednotlivé programově implementační detaily, ale pouze to, co jednotlivé části modelu dělají. Proto je podmínkou, aby výsledný model byl co nejpřehlednější a co nejjednodušší. Specifikace jednotlivých implementačních detailu je probrána v samotném návrhu. Pro prvotní analýzu se v rámci metody UP využívá diagram tříd z jazyku UML.

Při samotném vytváření diagramu se používají objekty skutečného světa např.: osoba, katalog, seminář atd. Hlavní podmínka ovšem musí být, že zmíněné pojmy jsou jasné a jednoznačné. Pokud tato podmínka není splněna je hlavním úkolem analýzy vytvořit takový model, který odstraní všechny nejasnosti. Jednotlivé objekty se modelují pomocí správné třídy a vytváří se jednotlivé závislosti mezi nimi. Samotné hledání analytických tříd pak probíhá následujících procesy.

1. Na základě analýzy podstatných jmen a sloves
2. Metodou štítků CRC (class, responsibilities & collabortors]
3. Hledání tříd z jiných zdrojů

Analýza podstatných jmen definuje jednotlivé třídy a atributy, kdežto pod jednotlivými slovesy jsou definovány samotné operace nebo odpovědnosti. Přestože je zmíněná definice jednoduchá, je potřeba si dávat pozor na synonyma a homonyma, které mohou vést k chybně definovaným třídám. V kombinaci se zmíněnou analýzou podstatných jmen se využívá metoda štítku CRC, která rozděluje nalezené třídy, atributy a operace do jednotlivých štítků (samolepících lístků). Jednotlivé štítky lze navzájem propojovat. Metodu štítku CRC lze zároveň kombinovat i s procesem hledání tříd z jiných zdrojů, ve které se analyzují objekty reálného světa např. norma, směrnice, zákon atd. Struktura těchto jednotlivých tříd byla čerpána z [17][18][19][20][21] Jednotlivé třídy tříd tedy představují velmi vysokou úroveň abstrakce a její název zpravidla odráží i její účel. Správná třída by celkově měla obsahovat menší počet odpovědností (doporučuje se mezi 3-5), které by měly být navzájem synergické. Jakmile je tento proces dokončen, lze začít modelovat diagram tříd.[2][3][5]

5.1 Analytický diagram tříd

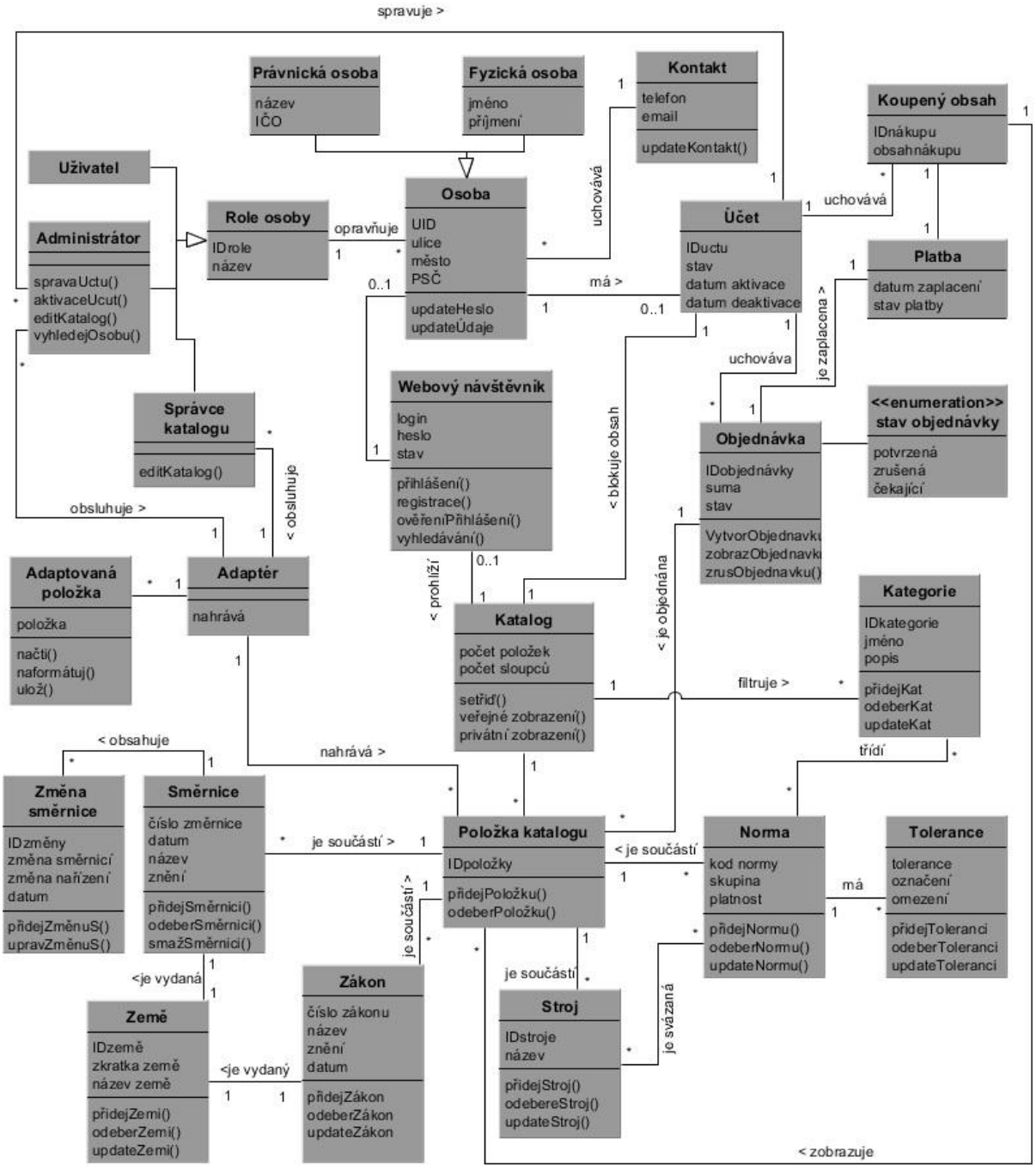
V následující kapitole je popsán samotný digram tříd, který je vytvořen za pomocí tříd a relací. Každá třída je definována pomocí atributů a operací a jednotlivé relace mají svojí násobnost a průchodnost. Kompletní analytický diagram je zobrazen na Obr. 15. Jednotlivé třídy byly i textově rozebrány a jsou obsažené v příloze DP. Pro ilustraci jsou popsány tři následující třídy.

Webový návštěvník - pokud na webové stránky dorazí návštěvník, může prohlížet části webu, aniž by byl přihlášený do IS. Nepřihlášený uživatel dostává omezené informace. Samotný systém dovoluje anonymní osobě využívat vyhledávač norem a zároveň má omezený přístup do katalogu norem. Anonymní návštěvník se může přihlásit a dle oprávnění osoby má různá omezení. Pokud není dosud registrován, může se i zaregistrovat.

Osoba - osoba se vytvoří s unikátním číslem a je v systému rozdělená na právnickou a

fyzickou osobu, pokud se registruje právnická osoba, musí být založena i fyzická osoba, která tu právnickou osobu zakládá. Registrovaná osoba bude muset uchovat základní informace o sobě (povinné: email, jméno a příjmení, název firmy a IČO). Osoby mají různé stupně oprávnění, které jsou definovány ve třídě role osoby, defaultně každá nová osoba bude uživatel. Osoba si pak může sama změnit heslo nebo údaje, k tomu slouží třída správa osoby. Osobě se musí zřídit účet, dokud nebude mít aktivní účet, bude stále dostávat omezené informace a nebude mít plný přístup.

Role osoby - tato třída představuje seznam rolí (oprávnění), které můžou mít osoby. Administrátorská práva dovolují aktivovat/deaktivovat účet osobám, spravovat údaje osob. Pomocí třídy správa norem bude přidávat normy přes webové rozhraní nebo používat adapter pro nahrávání norem do databáze.



Obr. 15 Analytický diagram tříd

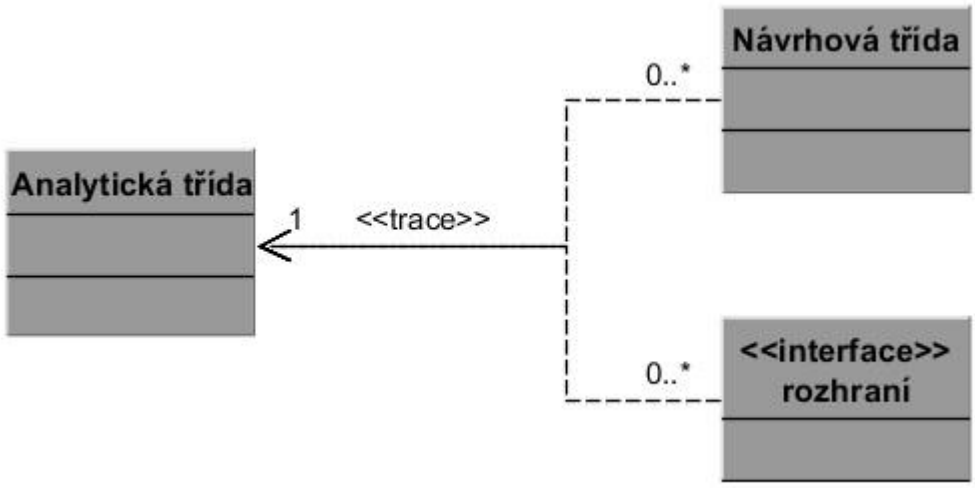
6 NÁVRH INFORMAČNÍHO SYSTÉMU

Tato kapitola popisuje samotný návrh IS, jehož cílem je konkrétní specifikace analytického modelu tak, aby ho bylo možné přímo implementovat. Zatímco analýza IS pracuje s pohledem samotných uživatelů systému, tak návrh IS se již zaměřuje na samotné řešení implementačního modelu. Pomyslná hranice mezi analytickým modelem a návrhovým modelem je velmi těsná, proto můžou být jednotlivé prvky návrhu obsaženy již v samotné analýze a výsledný návrhový model nemusí být o moc rozdílný než je v samotné analýze. V rámci návrhu se tedy pracuje s vytvořeným analytický modelem, který je postupně upřesňován, doplňují se datové typy a všechny stávající asociace jsou upřesněny na pomoci dvou specifických asociací

1. Agregace
2. Kompozice
3. Rozhraní

Agregace je tranzitivní a představuje volnější vazbu mezi objekty. Značí se čárou, která je na jedné straně zakončena prázdným kosočtvercem. Agregaci si lze představit jako sestavu/dílčí prvky. Pokud by sestava zanikla, tak jednotlivé dílčí prvky budou v systému zachovány. Pokud je nastavena jednostranná průchodnost na straně sestavy, dílčí prvky netuší o existenci sestavy. Příkladem může být barva a jakýkoliv objekt (dům, auto atd.). Pokud zruším auto, barva dál zůstává v systému.

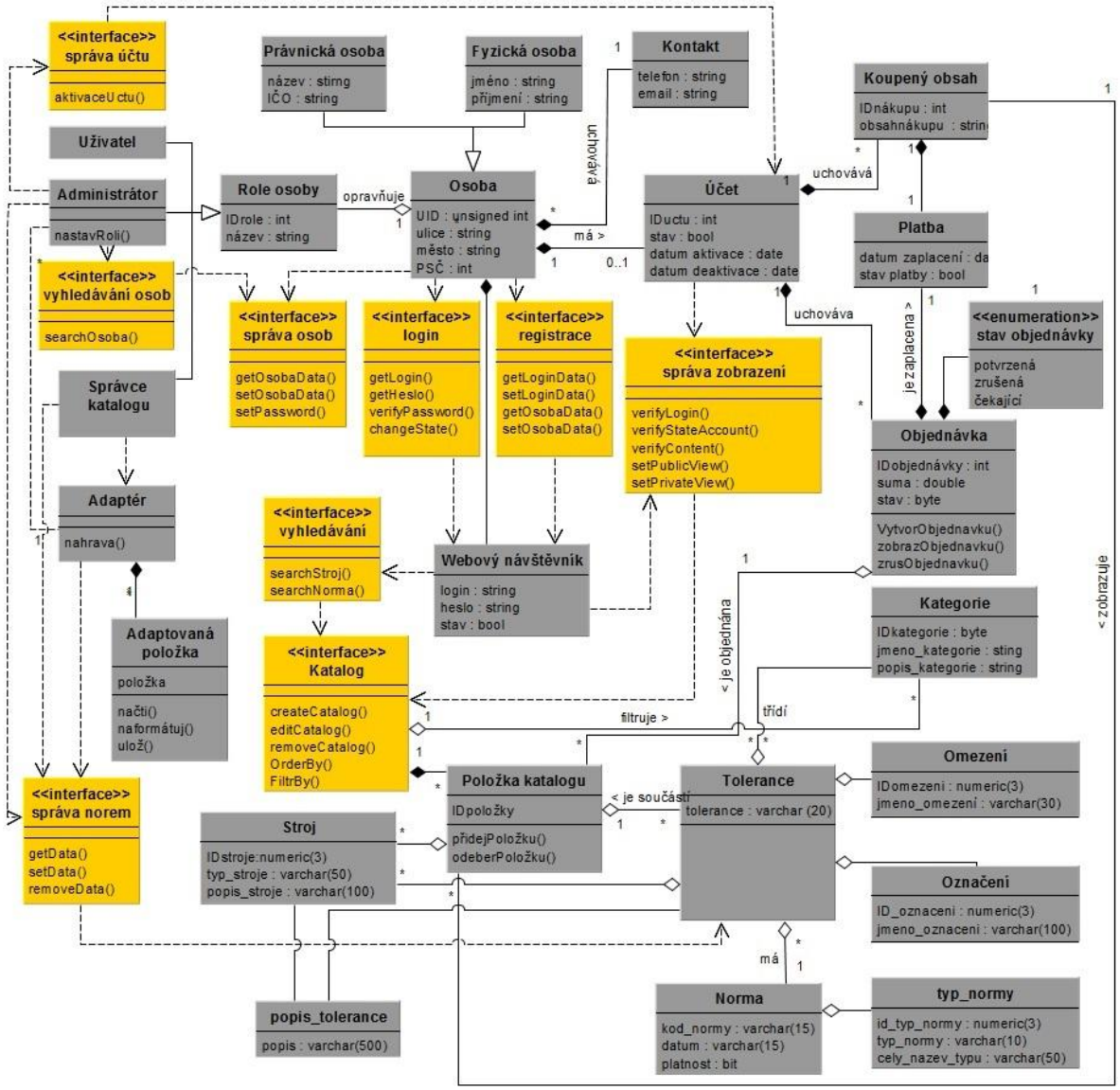
Kompozice představuje pevnější formu agregace a označuje se vyplněným kosočtvercem. Hlavní rozdíl je, že pokud zanikne sestava, tak dílčí prvek zanikne s ním. Pod tímto pojmem si lze představit např. knihu, strom atd. Pokud kniha zanikne, zanikají i její listy. Dále pak v systému nesmí zůstat žádná asociace typu *:* (N:M) z toho důvodu, že žádný programovací jazyk nedokáže takovou asociaci implementovat. Dále jsou přidána jednotlivá rozhraní, pomocí kterých jednotlivé části systému komunikují mezi sebou. Jedna analytická třída tedy může být rozdělena do několika návrhových tříd a rozhraní viz Obr. 16. [2][3][5][9]



Obr. 16 Rozložení obecnější analytické třídy do návrhových tříd a rozhraní [3]

6.1 Návrhový model diagramu tříd

Hlavní náplní je navrhnutí správné struktury databáze, aby redundance dat v databázi byla co nejmenší, tím je na mysli, že duplikace dat musí být minimální a byla zachována referenční integrita. Obecně holá data se rozdělují na logičtější celky. Tomuto procesu se říká normalizace. Z toho důvodu je v návrhovém modelu značně rozdělena třída norma. Jednotlivá data jsou rozdělena do tabulek a v rámci relační databáze se přistupuje k datům pomocí řídicích systému nepřímo skrz procesy. Kvůli tomu jsou metody ze třídy vyjmuty a ztvárněny jako rozhraní. Díky tomu jsou data stejně jako procesy nezávislá a můžeme využívat více zdrojů dat. Celý návrhový diagram je zobrazen na Obr. 17.



Obr. 17 Návrhový diagram informačního systému

7 IMPLEMENTACE INFORMAČNÍHO SYTÉMU

Samotná implementace návrhu informačního systému probíhá v jednotlivých následujících krocích počínaje vytvoření tabulek pomocí jazyku SQL. Propojení tabulek pomocí primárního klíče tj. unikátní identifikátor, který se už nesmí duplikovat znovu v tabulce a cizího klíče k propojení s jiným řádkem tabulky. Vytvoření jednotlivých procedur manipulujících se samotnými daty a nastavení triggerů pro omezující podmínky.

K samotné implementaci se používá Microsoft SQL Management studio express. Jedná se vývojové prostředí od Microsoftu a databáze se vytváří pro Microsoft SQL Server 2008 R2 [11]. Pomocí zmíněného prostředí lze buď databázi modelovat, nebo vytvářet jednotlivé částí přímo pomocí samotného SQL kódu. V jednotlivých následujících kapitolách jsou probrány jednotlivé části implementace.

7.1 Vytvoření databáze a tabulek

Základním krokem pro tvorbu databáze je vytvoření samotné databáze. Provádí následujícím jednoduchým SQL kódem, který je v Tab. 8.

Tab. 8 Ukázka kódu pro vytvoření databáze

```
1: CREATE DATABASE normy          //vytvoření databáze normy
```

Jakmile je vytvořena databáze, tak dalšími kroky je vytvoření tabulek. Jednotlivé tabulky představují základní kameny celé databáze, od kterých se všechny ostatní procedury odvíjí. Jednotlivé tabulky jsou zpracovány podle závislostí určených v návrhovém diagramu vytvořeném v kapitole 6. V jednotlivých tabulkách jsou definované datové typy, unikátnost a případně informace, zdali má něco obsahovat nebo ne. Kódy samotných tabulek jsou připojeny v příloze diplomové práce. Příklad kódu pro vytvoření tabulky je zobrazen v Tab. 9.

Tab. 9 Ukázka zdrojového kódu pro vytvoření tabulek

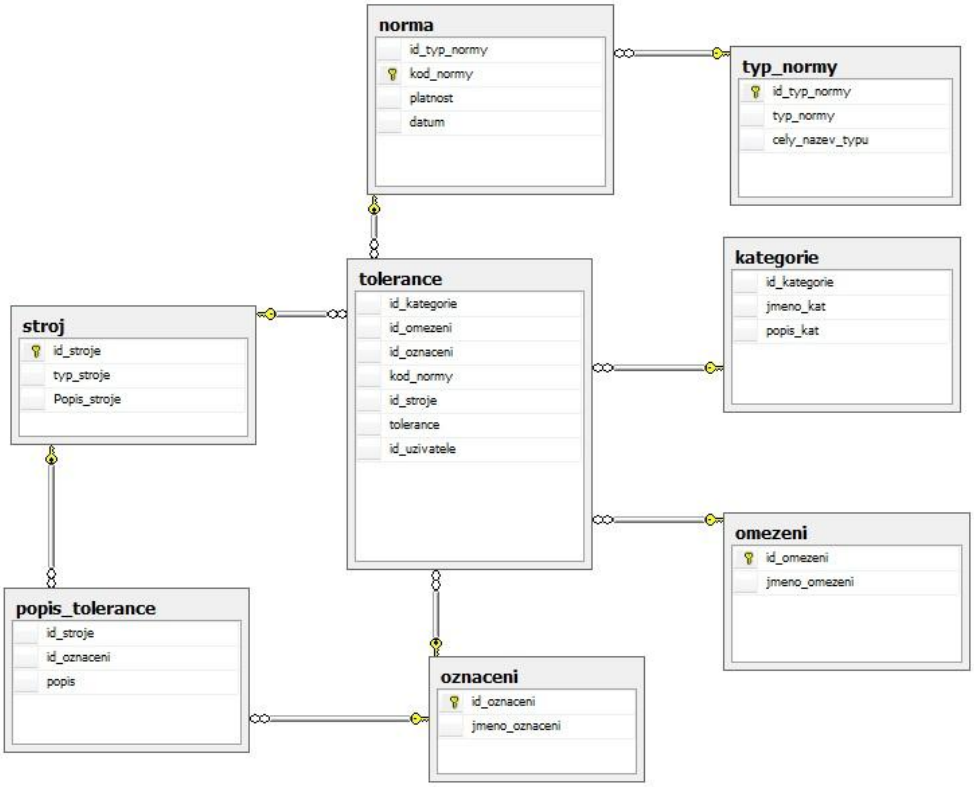
```
1: CREATE TABLE stroj                    //vytvoření tabulky stroj
2: (
3: id_stroje numeric(3) not null,        /*přidání různých řádku tabulky
4: typ stroje varchar(50) not null,      s různým datovým typem*/
5: popis_stroje varchar(50),
6: )
```

Jednotlivé tabulky jsou v relační databázi propojeny pomocí asociací a v implementaci se k propojení využívá primární a cizí klíče. Tyto klíče byly v tabulkách navrženy tak, aby rozvržení databáze splňovalo třetí normální formu. Jednotlivé klíče jsou v MS-SQL tvořeny pomocí následujícího kódu viz Tab. 10.

Tab. 10 Ukázka zdrojové kódu pro nastavení primárních a cizích klíčů

```
7:  alter table stroj                  //vybrání tabulky stroj
8:  add primary key (id_stroje)        //vytvoření primárního klíče na
9:                                     sloupci id_stroje
10: alter table popis_tolerance        //vybrání tabulky popis_tolerance
11: add foreign key (id_stroje)        //vytvoření cizího klíče ze stroje na
    references stroj(id_stroje)        popis_tolerance
```

Jakmile jsou jednotlivé klíče mezi sebou definovány lze si je zkontrolovat, zda implementace odpovídá návrhu. Část databáze a její propojení jednotlivých tabulek pomocí primárních a cizích klíčů je zobrazeno na obrázku Obr. 18.



Obr. 18 Propojení části tabulek pomocí primárních a cizích klíčů

Jednotlivý rozbor tabulek týkajících se norem a popis chování tabulky tolerance je rozepsán v následující Tab. 11.

Tab. 11 Rozbor jednotlivých tříd

Tabulka	**Popis**
Typ_normy	Typ_normy v sobě uchovává údaje o zkratkách norem tj. (ČSN, EN, DIN, BS atd.) a může uchovávat celé jméno zkratky. Typ normy je propojen cizím klíčem s tabulkou norma.
Norma	Norma uchovává číselný kód normy, na kterém je přiřazen primární klíč. Zároveň nese informaci o platnosti a datu normy.
Kategorie	Kategorie obsahuje seznam jednotlivých kategorií a jejich popis.
Omezení	Tabulka omezení obsahuje seznam jmen omezení (lineární osy, radiální osy atd.)
Označení	Označení obsahuje seznam názvu označení (EAX, EBX,Axial error atd.)
Stroj	Zaznamenává jednotlivé názvy strojů, jež mohou být rozšířeny o popis
Tolerance	Uchovává toleranci normy a odkazy na příslušné tabulky pomocí ID. Tyto odkazy jsou řešeny pomocí cizích klíčů, proto lze celou tabulku správným dotazem opětovně sestavit např. pomocí pohledu Tab. 12. Pro zabránění redundance se na vstupu musí současně hlídat unikátnost záznamu v tabulkách: Normy, Stroje, Omezení, Označení a Tolerance.
Popis	Uchovává popis označení. Popis označení ovlivňují tabulky označení a stroj.

Tolerance	Pro unikátnost záznamu se musí při vkládání záznamu kontrolovat obě tabulky zároveň.

7.2 Vytvoření pohledů

Hlavní předností pohledů je, že vytvořený pohled slouží pouze pro čtení a neumožňuje změnit data v databázi a pro různé uživatele lze vytvořit různé pohledy. Pohledy se tedy využívají kvůli bezpečnosti a v případě této diplomové práce je lze použít na webových stránkách jak pro přihlášené osoby, tak především pro neregistrované a nepřihlášené osoby. Ukázka zdrojového kódu pro vytvoření pohledu je zobrazena v Tab. 12

Tab. 12 Ukázka zdrojového kódu pro vytvoření pohledu

```
1:  CREATE VIEW PohledTolerance(NORMA, STROJ, OMEZENI, OZNACENI, TOLERANCE,
    POPIS, KATEGORIE)
2:  AS
3:  select (tn.typ_normy + t.kod_normy), s.typ_stroje, om.jmeno_omezeni,
    oz.jmeno_oznaceni, t.tolerance, p.popis, k.jmeno_kat
4:  FROM tolerance t, stroj s, omezeni om, oznaceni oz, kategorie k, norma
    n, typ_normy tn, popis_tolerance p
5:  WHERE t.kod_normy = n.kod_normy
6:  and t.id_stroje = s.id_stroje
7:  and t.id_omezeni = om.id_omezeni
8:  and t.id_oznaceni = oz.id_oznaceni
9:  and t.id_kategorie = k.id_kategorie
10 and oz.id_oznaceni = p.id_oznaceni
11 and s.id_stroje = p.id_stroje
12 and tn.id_typ_normy = n.id_typ_normy
```

7.3 Vytvoření procedur

Procedury v SQL představuje logickou skupinu příkazů, kterou lze následně volat jako funkce. Využívají se na místech, kde dochází k opakovanému použití souboru dotazu např. v bankovním sektoru při tisku smluv, kde se do smlouvy propisují osobní údaje, kontaktní údaje, údaje banky atd. Procedura postupně provádí dotazy, které obsahuje a po dokončení posledního příkazu vrací teprve výsledek. To má za následek zrychlení celé databáze. Uživatelské rozhrání jenž slouží k ovládání samotné databáze, nemusí obsahovat metody s SQL dotazy, ale stačí pouze volat příslušné procedury pro práci s databází, které jsou již implementovány v rámci serveru.

Procedury je možné použít pro jednotlivé příkazy select, kde pouze přebíráme jednu proměnnou a vracíme výsledek podle údaje, který dostaneme, viz Tab. 13.

Tab. 13 Procedura pro vyhledávání normy podle stroje

```
1:  CREATE PROCEDURE [dbo].[VyhledaniNormyPodleStroje]
2:  @stroj varchar(50)
3:  AS
4:  SELECT typ_normy.typ_normy, norma.kod_normy, norma.platnost
5:  FROM stroj INNER JOIN (tolerance INNER JOIN (norma INNER JOIN typ_normy ON
6:  norma.id_typ_normy = typ_normy.id_typ_normy)
7:  ON tolerance.kod_normy = norma.kod_normy)
8:  ON tolerance.id_stroje = stroj.id_stroje
9:  WHERE typ_stroje = @stroj
```

Anebo lze proceduru využít pro desítky jednotlivých příkazu. V následující Tab. 14 je

zobrazená část kódu procedury pro plnění jednotlivých tabulek. Systém požádá o všechny proměnné. Jakmile je získá, tak začne postupně ukládat data do databáze. Procedura má v sobě navíc definovanou transakci, která v případě chybného zápis proces ukládání přeruší a vrátí databázi do předchozího stavu. O správnost formátu dat, se starají triggery (spouštěče).

Tab. 14 Ukázka zdrojového kódu pro vytvoření procedury

```
1:  CREATE PROCEDURE [dbo].[setData]
    /*deklarovani místních proměnných*/
2:  @jmeno_katIN varchar(100),
3:  @typ_normyIN varchar(10),
4:  @strojIN varchar(50),
    …
5:  @kod_normyID varchar(15) output,
6:  @typ_normyID numeric(3) output,
7:  @strojID numeric(3) output,
    …
8:  AS
9:  /* začátek transakce, všechny podmínky uvnitř transakce musí být
10: splněny*/
11: BEGIN TRANSACTION
12: /* pridani stroje (pokud neexistuje) a nacteni id*/
13: if exists (SELECT typ_stroje from stroj WHERE typ_stroje = @strojIN )
14:       begin
15:             SELECT @strojID = id_stroje FROM stroj
16:             WHERE typ_stroje = @strojIN
17:       end
18: else
19:       begin
20:       SELECT @strojID = MAX(id_stroje) FROM stroj
21:       set @strojID = @strojID + 1
22:       INSERT INTO stroj VALUES
23:       (@strojID,@strojIN, null)
24: end
    …

    /*kontrola vkládaného záznamu na redundanci*/
25: if not exists
26:       (SELECT * FROM tolerance
27:       WHERE
28:       id_kategorie = @kategorieID
29:       AND id_omezeni = @omezeniID
30:       AND id_oznaceni = @oznaceniID
31:       AND kod_normy = @kod_normyID
32:       AND id_stroje = @strojID
33:       AND tolerance = @tolerance)
34:
35:       BEGIN                 //vložení záznamu pokud neexistuje záznam
36:             INSERT INTO tolerance VALUES
37:             (@kategorieID, @omezeniID, @oznaceniID, @kod_normyID,
38: @strojID, @tolerance, null)
39:       end
40: else
41:       begin
42:       print 'V databazi je uz záznam'
          end
    …
43: COMMIT //potvrzení transakce
```

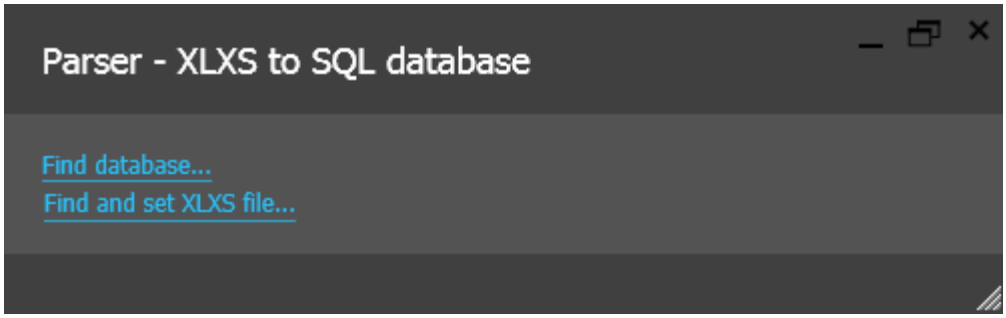
7.4 Adaptér pro import dat do databáze

Hlavním úkolem adaptéru je jednoduché naplnění databáze již předpřipravenými daty. Pro vývoj této aplikace byl vybrán jazyk C#, protože obsahuje třídy, které jsou připraveny pro efektivní komunikaci s databází za využití uložených procedur na straně serveru. Data do tabulek databáze jsou importována z .xlsx souborů. Jazyk C# disponuje již připraveným rozhraním pro načítání dat z .xlsx. Aplikace je schopna načíst data z příslušného .xlsx souboru a těmito daty naplnit příslušné tabulky. Ukázka volání procedury je zobrazeno v Tab. 15.

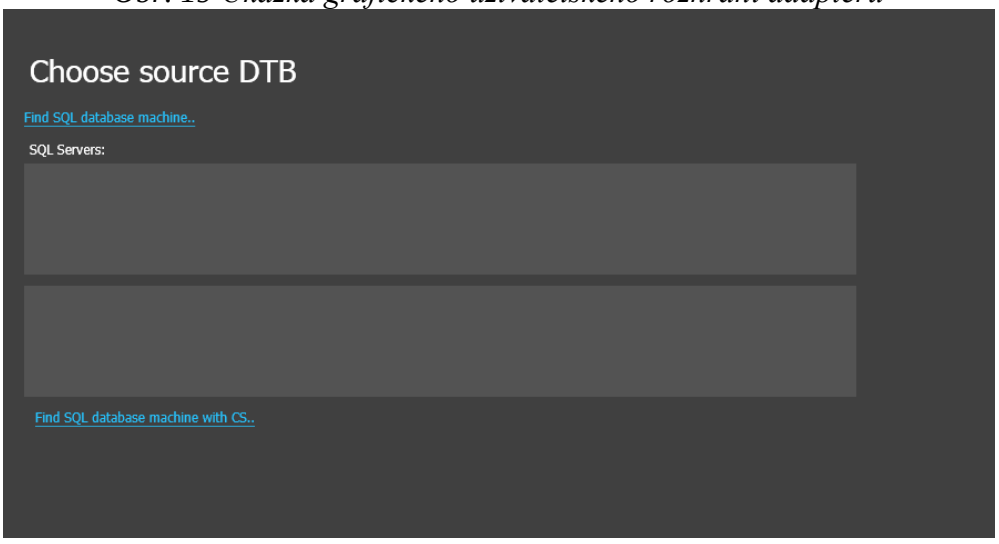
Tab. 15 Obecné příkazy pro obsluhu procedur [15]

```
1: SqlCommand command = new SqlCommand("ProcedureName",sqlConnection);
2: command.CommandType=CommandType.StoreProcedure;
3: command.Parameters.AddWithValue("@variable",value);
4: command.ExecuteNonQuery();
```

Výsledná podoba adaptéru zobrazena na Obr. 19. Ve které jsou na výběr dvě možnosti. První možnost představuje vybrání samotné databáze, kterou je potřeba naplnit daty viz Obr. 20. a druhá vybrání zdrojového .xlxs souboru.



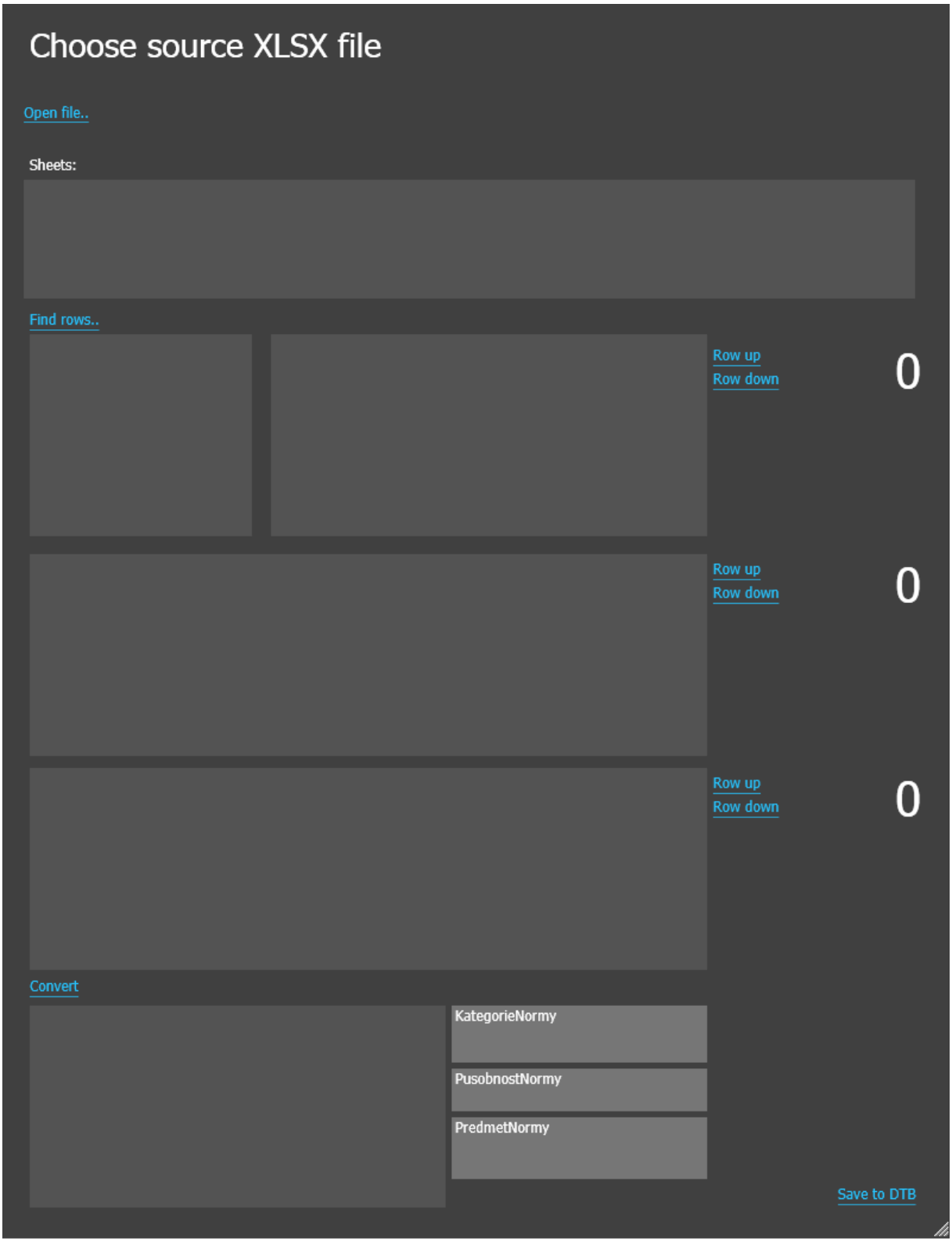
Obr. 19 Ukázka grafického uživatelského rozhraní adaptéru



Obr. 20 Ukázka grafického rozhraní při výběru SQL Serveru

Jakmile je vybrána databáze, následujícím krokem je načtení .xlsx souboru. Samotný .xlsx soubor obsahuje několik listů (sheets), kde každý list je definován jako jedna kategorie normy. Obsah jednotlivých kategorií je zobrazen v Tab. 2. Každá list pak obsahuje jednotné uspořádání dat pro samotné normy (viz Tab. 3).

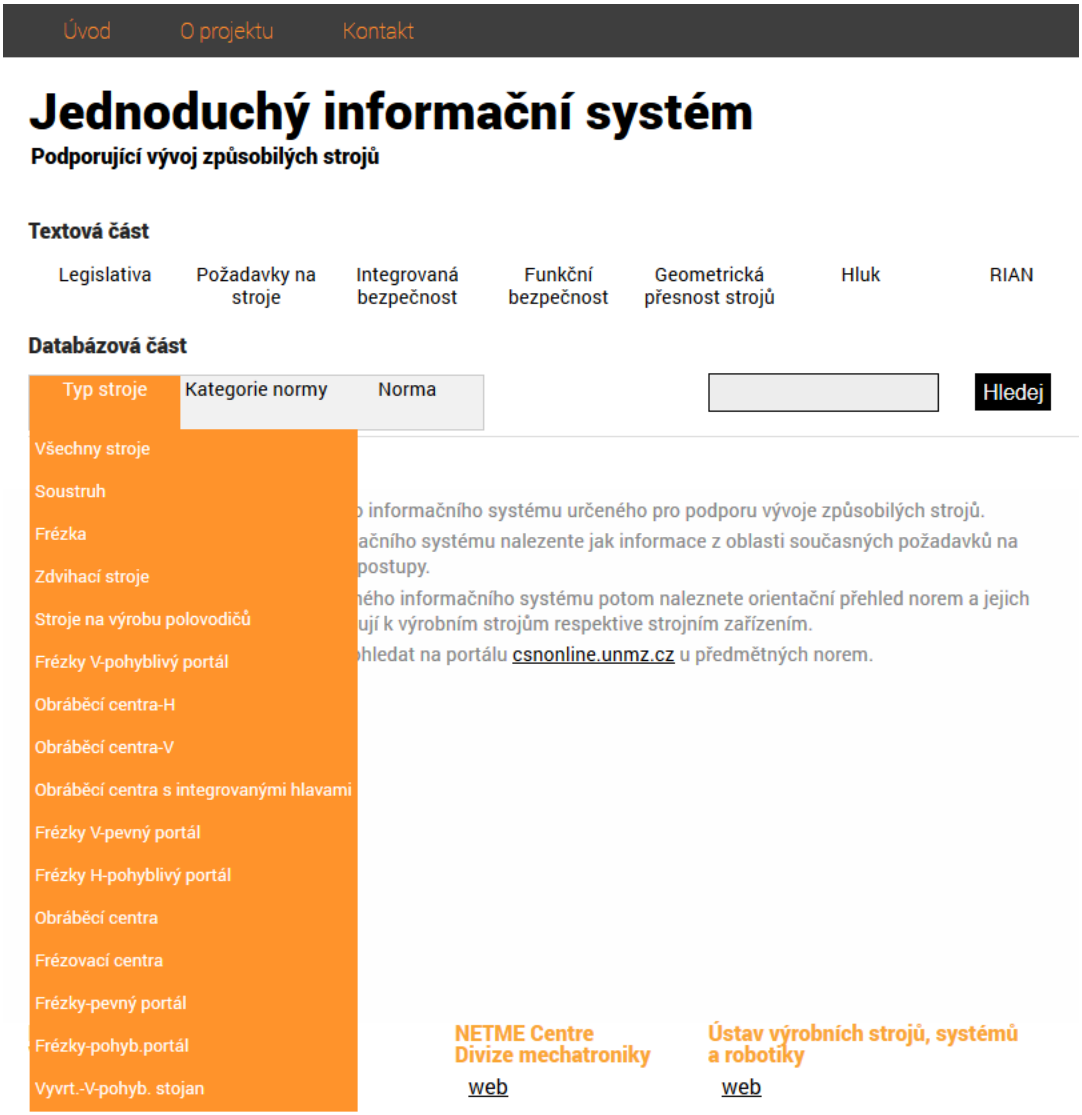
Adaptér po vybrání souboru načte jednotlivé listy a vybraný list postupně proskenuje. Získaná data ze souboru jsou uložena do proměnných a předány proceduře, která postupně naplní jednotlivé tabulky. Grafické rozhraní pro výběr souboru je zobrazeno na Obr. 21.



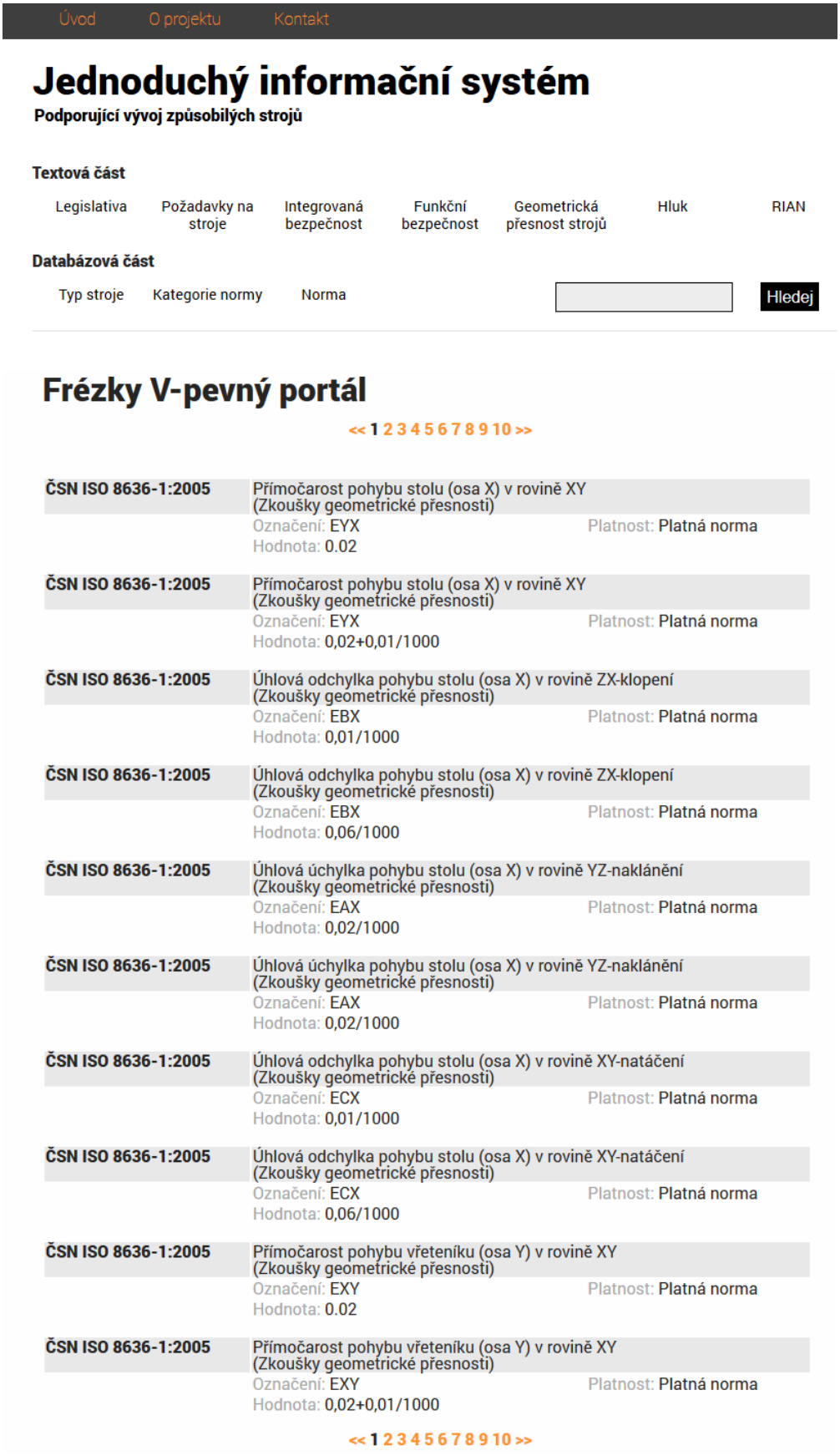
Obr. 21 Adaptér – načtení dat z .xlsx souboru

7.5 Jednoduchý informační systém

Náplní této práce není vytvoření samotného webu informačního systém, ale pro ukázku jak samotné databáze pracují s webovým rozhraním, je znázorněn jednoduchý informační systém na následujících obrázcích (Obr. 22 a Obr. 23). Na Obr. 22 je zobrazen seznam všech strojů obsažených v databázi. Na Obr. 23 je pak zobrazen výstup všech norem, které jsou v databázi připojené ke stroji „Frézky V-pevný portál“. Následující obrázky byly poskytnuty se souhlasem vedoucího diplomové práce, který je zároveň i jejich autorem.



Obr. 22 Ukázka funkčního webového rozhraní zobrazující přehled strojů



Obr. 23 Ukázka zobrazující detail jednotlivých norem

8 ZÁVĚR

Tato diplomová práce se zabývá analýzou, návrhem a implementací informačního systému. Touto problematikou se v praxi nezabývá pouze jedna osoba, ale pracuje zde celý tým skládající se z metodiků, analytiků, designérů, programátorů a testerů.

Práce obsahuje rešeršní část v rámci třetí kapitoly, kde je proveden podrobný rozbor přístupu pro vývoj informačních systémů. Na základě této rešerše byla zvolena objektová metoda UP pro postup vývoje řešeného IS. Samotná metoda UP pracuje se standardizovaným jazykem UML. Tento jazyk pro jednotlivé etapy vývoje využívá specifické typy diagramů.

Prvním krokem vývoje IS je definice požadavků, které jsou na něj kladeny. Definice správných požadavků na systém je stěžejní částí vývoje IS, protože následující etapy vývoje jsou přímo závislé právě na těchto atributech. Na základě jednotlivých požadavků byl vytvořen tzv. use case model a zároveň byly vytvořeny části grafického uživatelského rozhraní budoucího IS. Tuto problematiku řeší čtvrtá kapitola.

V další etapě vývoje byla provedena analýza IS za pomoci diagramu tříd, která specifikuje objektovou strukturu databáze nezávisle na zvolené platformě. Analytický model je abstraktní koncept databáze. Tato abstrakce se v dalších etapách vývoje postupně snižuje specifikováním jednotlivých modelů. Tuto problematiku řeší pátá kapitola.

Šestá kapitola se zabývá konečným návrhem databáze. Výsledkem této etapy je návrhový model ve formě diagramů tříd, jenž slouží pro konečnou implementaci databáze. Tato fáze je již závislá na zvolené platformě. Jedním z bodů zadání bylo využití jazyka SQL, a proto byl v rámci této etapy zvolen relační typ databáze. Jednotlivé třídy byly rozvrženy do třetí normální formy a to z důvodu snížení redundance dat.

Další etapou vývoje IS je jeho praktická realizace. Tato diplomová práce není zaměřená na vývoj IS jako celku, ale pouze na vývoj samotné databáze. V rámci realizace byly na základě návrhového diagramu tříd a pravidel relační databáze vytvořeny tabulky a relace databáze. Pro manipulaci s daty v rámci databáze byly navrženy procedury za pomocí jazyka Transact-SQL. Hlavní výhodou implementace těchto procedur je zefektivnění vývoje uživatelské aplikace, jejíž vývojář nemusí vytvářet dotazy přímo v kódu aplikace, ale může přímo volat připravené procedury. Praktická implementace databáze je popsána v sedmé kapitole.

Hlavním přínosem této diplomové práce je seznámení se s řadou metod vývoje IS, které se používají v různých odvětvích softwarového inženýrství. Dalším přínosem bylo seznámení se s tvorbou samotné databáze, jazykem SQL a jeho procedurálním rozšířením.

SEZNAM POUŽITÉ LITERATŮRY

[1] KAJZAR, Dušan a Ivan POLÁČEK. *Projektování informačních systémů I: strukturovaný a objektový přístup*. Opava, 2003. ISBN 80-7248-214-9.

[2] KRAVAL, Ilja. *Analytické modelování informačních systémů pomocí UML v praxi*. Lipina: Object Consulting, 2010. ISBN 978-80-254-6986-6.

[3] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací :objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[4] SOMMERVILLE, Ian. *Softwarové inženýrství.* Brno: Computer Press, 2013. ISBN 978-80-251-3826-7.

[5] KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. Brno: Computer Press, 2004. ISBN 80-251-0231-9.

[6] DEWSON, Robin. *Beginning SQL Server 2008 for Developers: From Novice to Professional*. USA: Apress. ISBN-13 978-1-59059-958-7.

[7] ARIE D. Jones, Ryan K. STEPHENS a Ron PLEW. *Naučte se SQL za 28 dní: Stačí hodina denně*. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

[8] BECK, Kent a Pavel MAKOVEC. *Extrémní programování: knihovna programátora*. Praha: Grada Publishing, 2002. ISBN 80-247-0300-9.

[9] ILJA, Kraval. *Objektové modelování pomocí UML v praxi: 2005* [online]. 2005 [cit. 2014-05-30]. Dostupné z:http://www.objects.cz

[10] HRUŠKA, Tomáš a Zbyněk KŘIVKA. FIT VUT BRNO. *Informační systémy (IIS, PIS)*. Brno, 2007.

[11] MISTRY, Ross a Stacia MISNER. *Introducing Microsoft SQL Server 2008 R2* [online]. 2010 [cit. 2014-01-10]. Dostupné z: http://blogs.msdn.com/b/microsoft_press/archive/2010/04/14/free-ebook-introducing-microsoft-sql-server-2008-r2.aspx

[12] Oddělení vývoje systémových služeb: SCRUM. *Masarykova univerzita: ústav výpočetní techniky* [online]. 2014 [cit. 2014-05-20]. Dostupné z: http://www.muni.cz/ics/925600/web/scrum

[13] ROYCE, Winston. *Managing the Development of Large Software Systems* [online]. Proceedings of IEEE WESCON, 1970 [cit. 2014-05-20]. Dostupné z: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[14] VOLODARSKY, Mike. MSDN Magazine. MICROSOFT. *IIS 7.0: Explore The Web Server For Windows Vista And Beyond* [online]. 2007 [cit. 2014-03-20]. Dostupné z:http://msdn.microsoft.com/en-us/magazine/cc163453.aspx

[15] Configuring Parameters and Parameter Data Types: .NET Framework 4.5. MICROSOFT. *Developer Network*[online]. 2014 [cit. 2014-05-25]. Dostupné z: http://msdn.microsoft.com/en-us/library/yy6y35y8(v=vs.110).aspx

[16] Co znamená zkratka ISO a další informace. MBK CONSULTING. *MBK* [online]. 2013 [cit. 2014-02-15]. Dostupné z: http://www.mbk.cz/iso/co-znamena-zkratka-iso-a-dalsi-informace

[17] Často kladené otázky - Technická normalizace. *Úřad pro technickou normalizaci, metrologii a státní zkušebnictví.* [online]. 2014 [cit. 2014-02-15]. Dostupné z: http://www.unmz.cz/urad/casto-kladene-otazky-technicka-normalizace

[18] Normy EU. *Ministerstvo zahraničních věcí ČR* [online]. 2014 [cit. 2014-02-15]. Dostupné z: http://www.mzv.cz/representation_brussels/cz/evropska_unie/eu_pro_podnikatele/jak_ovlivnit_predpisy_eu/normy_eu/index.html

[19] CS NORMY. *CS NORMY* [online]. 2014 [cit. 2014-02-15]. Dostupné z: http://www.csnormy.cz/

[20] ISO. *ISO* [online]. 2014 [cit. 2014-03-15]. Dostupné z: http://www.iso.org/iso/home/about.htm

[21] Směrnice – definice. *Evropská komise* [online]. 2012 [cit. 2014-03-15]. Dostupné z:http://ec.europa.eu/eu_law/directives/directives_cs.htm

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH NA DVD

Přiložené DVD obsahuje:

1: Diplomovou práci ve formátu .pdf
2: Přehled případů užití v textové verzi .docx
3: Diagram případů užití ve formátu .graphml
4: Analytický diagram tříd vytvořený ve formátu .graphml
5: Návrhový diagram tříd ve formátu .graphml
6: Kódy pro vytvoření databáze ve formátu .sql

Soubory .graphml představují grafy vytvořené pomocí free programu yEd-GrphEditor a lze jej stáhnout zdarma ze stránek výrobce viz http://www.yworks.com/en/products_yed_download.html, kde jsou jednotlivé verze pro různé systémy.

K otevření souboru sql je zapotřebí použít SQL Server Managment Studio, které lze stáhnout zdarma ve verzi Express z webových stránek Microsoftu.